ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΈΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή παρακολούθησης χρήσης κινητού τηλεφώνου σε πλατφόρμα Android.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Β. Κοζομπόλης

Αθήνα, Ιούλιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή παρακολούθησης χρήσης κινητού τηλεφώνου σε πλατφόρμα Android.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Β. Κοζομπόλης

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ..... .

.....	.....	.....
Ευστάθιος Συκάς	Μιχαήλ Θεολόγου	Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2016

.....
Νικόλαος Β. Κοζομπόλης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής, για έξυπνο κινητό τηλέφωνο (smartphone) Android, που να παρέχει στο χρήστη την δυνατότητα παρακολούθησης της κατανάλωσης του προπληρωμένου χρόνου ομιλίας, γραπτών μηνυμάτων και δεδομένων διαδικτύου. Η εφαρμογή παρακολουθεί τοπικά την χρήση και ανά διαστήματα επαληθεύει τα δεδομένα με τα αντίστοιχα του παρόχου, αποστέλλοντας αίτημα, υπό μορφή αυτοματοποιημένου γραπτού μηνύματος. Η διεπαφή της εφαρμογής υλοποιήθηκε έτσι ώστε να παρέχει στον χρήστη την δυνατότητα διαρκούς εποπτείας του υπολοίπου, με την καταβολή ελάχιστης φυσικής και νοητικής προσπάθειας εκ μέρους του.

Λέξεις Κλειδιά

Android, Διεπαφή, Αλληλεπίδραση Ανθρώπου-Υπολογιστή, Προπληρωμένο Πακέτο, Χρήστης, Κινητό Τηλέφωνο, Smartphone, Πάροχος.

Abstract

The purpose of this thesis, titled "Android smartphone application for utilization monitoring", is the development of an application for Android smartphones that audits the consumption of prepaid voice calls, sms and mobile data on behalf of the user. The application monitors the usage locally and , at certain intervals, confirms the measurements by sending an automated text message to the provider. The graphic interface of the application was designed as to provide uninterupted balance monitoring and minimize the customer's physical and mental effort.

Key words

Android, Interface, Human-Computer Interaction, Prepaid Bundle, User, Mobile Phone, Smartphone, Provider.

Πίνακας περιεχομένων

Εισαγωγή

Σκοπός αυτής της εργασίας είναι η ανάπτυξη μιας εφαρμογής για έξυπνο κινητό τηλέφωνο (smartphone) Android, που να παρέχει την δυνατότητα παρακολούθησης της κατανάλωσης του προπληρωμένου πακέτου (χρόνου ομιλίας, γραπτών μηνυμάτων και δεδομένων), με την καταβολή ελάχιστης φυσικής και νοητικής προσπάθειας εκ μέρους του χρήστη.

Στο **Κεφάλαιο 1 (Περιγραφή του προβλήματος και προτεινόμενη λύση)** παρουσιάζονται οι τρόποι τιμολόγησης των τηλεπικοινωνιακών υπηρεσιών, οι τύποι των Υπηρεσιών Ενημέρωσης Υπολοίπου που χρησιμοποιούν οι πάροχοι και εντοπίζονται οι τομείς που επιδέχονται βελτίωσης, ως προς την εύκολη και επακριβή ενημέρωση του συνδρομητή. Στη συνέχεια προτείνεται συγκεκριμένη λύση και αναφέρονται τα τεχνικά χαρακτηριστικά της.

Στο **Κεφάλαιο 2 (Λειτουργικό σύστημα Android)** παρουσιάζονται οι δυνατότητες και η αρχιτεκτονική του λειτουργικού συστήματος Android.

Στο **Κεφάλαιο 3 (Επικοινωνία Ανθρώπου – Υπολογιστή)** γίνεται αναφορά στην έννοια της Αλληλεπίδρασης Ανθρώπου – Υπολογιστή, παρουσιάζονται οι Αρχές Σχεδιασμού Διεπαφών Οθόνης και περιγράφεται η εφαρμογή τους στην υλοποίηση της διεπαφής.

Στο **Κεφάλαιο 4 (Οι Όψεις της Εφαρμογής)** παρουσιάζονται οι όψεις της εφαρμογής και ο τρόπος μετάβασης από την μια όψη στην άλλη.

Στο **Κεφάλαιο 5 (Ακρίβεια Δεδομένων)** αναλύεται το σημαντικό πρόβλημα της μη ταύτισης των τιμών των μετρουμένων μεγεθών (υπόλοιπο χρόνου ομιλίας, γραπτών μηνυμάτων, δεδομένων) μεταξύ του πελάτη και του παρόχου και παρουσιάζεται ο τρόπος αντιμετώπισής του, στο πλαίσιο της εφαρμογής.

Στο **Κεφάλαιο 6 (Σχολιασμός Δομικών Στοιχείων της Εφαρμογής)** επεξηγούνται οι επί μέρους λειτουργίες του κώδικα java, του γραφικού περιβάλλοντος και του manifest.της εφαρμογής.

Στο Στο **Κεφάλαιο 7** παρουσιάζονται οι **Προτάσεις για βελτίωση** της εφαρμογής, ενώ το σύνολο του κώδικα επισυνάπτεται στο Παράρτημα 'Α'.

Κλείνοντας αυτή τη συνοπτική εισαγωγή της διπλωματικής εργασίας σημειώνεται ότι η εφαρμογή που αναπτύχθηκε φέρει την ονομασία 'Trindicator', που προκύπτει από την σύνθεση των λέξεων triple indicator (= τριπλός ενδείκτης), λόγω της δυνατότητας που έχει να παριστά ταυτόχρονα το υπόλοιπο προπληρωμένου χρόνου ομιλίας, γραπτών μηνυμάτων και δεδομένων.

Κεφάλαιο 1

Περιγραφή του προβλήματος και προτεινόμενη λύση.

Στο Κεφάλαιο αυτό παρουσιάζονται οι τρόποι τιμολόγησης των τηλεπικοινωνιακών υπηρεσιών, οι τύποι των Υπηρεσιών Ενημέρωσης Υπολοίπου που χρησιμοποιούν οι πάροχοι και εντοπίζονται οι τομείς που επιδέχονται βελτίωσης, ως προς την ευκολία και ακρίβεια ενημέρωσης του συνδρομητή. Στη συνέχεια προτείνεται συγκεκριμένη λύση και αναφέρονται τα τεχνικά χαρακτηριστικά της.

1.1 Γενικά

Οι πάροχοι κινητών τηλεπικοινωνιών για τις υπηρεσίες φωνητικών κλήσεων, μηνυμάτων και δεδομένων που προσφέρουν στους πελάτες τους εφαρμόζουν συνήθως μια από τις παρακάτω μεθόδους πληρωμής:

Μεταπληρωμή (postpay). Πρόκειται για τον συμβατικό τρόπο χρέωσης κατά τον οποίο οι πελάτες κάνουν χρήση των παρεχομένων υπηρεσιών, για συγκεκριμένο χρονικό διάστημα, συνήθως ενός μηνός. Ο πάροχος τηρεί στοιχεία χρήσης για έκαστο των πελατών, τα οποία τιμολογεί στο τέλος του μηνός και τα κοινοποιεί στον πελάτη για εξόφληση.

Προπληρωμή (prepay). Πρόκειται για μηχανισμό τιμολόγησης, όπου ο πελάτης προπληρώνει συγκεκριμένο πακέτο της υπηρεσίας που τον ενδιαφέρει και κατόπιν αρχίζει να την χρησιμοποιεί. Συνήθως ο πελάτης δεν λαμβάνει κανένα τιμολόγιο και χρεώνεται σε πραγματικό χρόνο από τα υψηλής διαθεσιμότητας συστήματα τιμολόγησης. Όταν το προπληρωμένο πακέτο εξαντληθεί τότε ο πελάτης χρεώνεται αναλόγως της χρήσης (π.χ ανά λεπτό για τις φωνητικές κλήσεις, ανά μήνυμα, ανά Megabyte δεδομένων) ή εναλλακτικά μπορεί να αγοράσει επιπλέον προπληρωμένο πακέτο. Σημειώνεται ότι η χρέωση αναλόγως της χρήσης είναι συνήθως ακριβότερη του προπληρωμένου πακέτου.

Γίνεται φανερό ότι στην περίπτωση του προπληρωμένου πακέτου ο πελάτης ενδιαφέρεται για την παρακολούθηση της κατανάλωσης προκειμένου να ελέγξει τη χρήση του ή να προβεί έγκαιρα στην αγορά επιπλέον πακέτου. Για το λόγο αυτό οι πάροχοι διαθέτουν ανάλογες Υπηρεσίες Ενημέρωσης Υπολοίπου (ΥΕΥ), όπως παρακάτω:

- **Υπηρεσία Διαδραστικής Φωνητικής Απόκρισης**- ΔΦΑ (Interactive Voice Response – IVR), όπου ο πελάτης αφού καλέσει ένα βραχύ αριθμό, μπορεί να ακούσει το

υπόλοιπο του προπληρωμένου πακέτου.

- **Υπηρεσία Σύντομου Μηνύματος** (SMS), που λειτουργεί όπως η παραπάνω ΔΦΑ, αλλά μέσω SMS.
- **Κέντρο Κλήσης** (Call Center) όπου ο πελάτης μπορεί να καλέσει το τηλεφωνικό κέντρο και να πληροφορηθεί το υπόλοιπο του πακέτου.
- **Online**, όπου ο πελάτης μπορεί να επισκεφθεί το λογαριασμό στην πύλη (portal) του παρόχου και να ενημερωθεί για την κατανάλωση του πακέτου.
- **Εφαρμογές**, που δίνουν τη δυνατότητα παρακολούθησης της χρήσης της συσκευής και παρέχουν στον πελάτη σχετικές πληροφορίες.
- **Ειδοποιήσεις SMS**, με τις οποίες ο πάροχος ενημερώνει τους πελάτες όταν η κατανάλωσή τους έχει υπερβεί ένα ορισμένο ποσοστό του πακέτου (π.χ. 80%, 100%, κ.λπ.)

Τα βασικά κίνητρα των παρόχων για την ανάπτυξη και υποστήριξη των Υπηρεσιών Ενημέρωσης Υπολοίπου είναι τα εξής:

- Η κάλυψη της ανάγκης πληροφόρησης των πελατών για τον βαθμό κατανάλωσης του προπληρωμένου πακέτου. Σημειώνεται ότι έχει διαπιστωθεί δυσκολία των πελατών να ελέγχουν υπηρεσίες, όπως τα δεδομένα, ως προς την ποσότητα των megabytes που δαπανήθηκαν. Αυτό τους καθιστά επιφυλακτικούς στο να εξαντλήσουν το προπληρωμένο όριο και προκαλείται η αρνητική εντύπωση ότι η υπηρεσία είναι ακριβή.
- Η αποφυγή συνέχισης της χρήσης μετά την εξάντληση του προπληρωμένου πακέτου, γεγονός που συνεπάγεται αυξημένες χρεώσεις οι οποίες προκαλούν δυσφορία στον πελάτη, όταν τελικά το διαπιστώσει.

1.2 Περιγραφή του προβλήματος.

Σε σχέση με την λειτουργικότητα και την ευχρηστία των Υπηρεσιών Ενημέρωσης Υπολοίπου, όπως αυτές θεωρούνται από την πλευρά του χρήστη, σημειώνεται ότι:

- Η πρόσβαση στις Υπηρεσίες Ενημέρωσης Υπολοίπου απαιτεί την καταβολή προσπάθειας από την πλευρά του πελάτη ώστε να ολοκληρώσει με επιτυχία τους κατάλληλους χειρισμούς για σύνδεσή του σε αυτή που επιθυμεί. Οι απαιτούμενοι χειρισμοί μπορεί να είναι απλοί όπως π.χ το άνοιγμα μιας εφαρμογής ενημέρωσης υπολοίπου και η ανάγνωση των αποτελεσμάτων ή μπορεί να είναι πλέον σύνθετοι, όπως η σύνταξη ενός μηνύματος SMS, η αποστολή του στον κατάλληλο αριθμό και η

ανάγνωση του μηνύματος απάντησης. Ανεξαρτήτως όμως της απαιτούμενης μικρής ή μεγαλύτερης προσπάθειας, αποτελούν ένα νοητικό φορτίο για τον χρήστη και προϋποθέτουν την, έστω και για μικρό χρονικό διάστημα, εστίαση της προσοχής του στον επιδιωκόμενο σκοπό.

- Οι Υπηρεσίες Ενημέρωσης Υπολοίπου που προαναφέρθηκαν παρέχουν στον πελάτη ένα στιγμιότυπο του υπόλοιπου προπληρωμένου πακέτου, για την στιγμή που υποβλήθηκε το ερώτημα. Οι καταναλώσεις του προπληρωμένου πακέτου που πραγματοποιούνται μετά την υποβολή του ερωτήματος θα κοινοποιηθούν στον πελάτη όταν αυτός υποβάλει νέο ερώτημα. Σημειώνεται ότι, όταν το προπληρωμένο πακέτο τείνει να εξαντληθεί, οι χρήστες έχουν την τάση να ερωτούν συχνότερα σχετικά με το διαθέσιμο υπόλοιπο, προκειμένου να αποφύγουν τις εκτός πακέτου χρεώσεις, που είναι μεγαλύτερες.

Μετά τα παραπάνω γίνεται φανερό ότι υπάρχει περιθώριο βελτίωσης της ευκολίας πρόσβασης στις Υπηρεσίες Ενημέρωσης Υπολοίπου καθώς και η ανάγκη παροχής στους πελάτες της δυνατότητας παρακολούθησης του υπόλοιπου με ακρίβεια, χωρίς την διαρκώς επαναλαμβανόμενη κλήση της Υπηρεσίας Ενημέρωσης Υπολοίπου.

1.3 Προτεινόμενη λύση.

Η προτεινόμενη λύση παρέχει στον χρήστη μια απλούστερη, ακριβέστερη και εύκολα προσπελάσιμη μέθοδο για την παρακολούθηση της κατανάλωσης του προπληρωμένου πακέτου, με χρήση συσκευών smartphones. Τα υπόλοιπα χρόνου ομιλίας, γραπτών μηνυμάτων και δεδομένων απεικονίζονται υπό μορφή ραβδογραμμάτων στο αρχικό μενού της συσκευής, με τρόπο ανάλογο εκείνου της στάθμης ισχύος σήματος και του βαθμού φόρτισης του συσσωρευτού (μπαταρίας). Αυτό εξασφαλίζει στον χρήστη, με μια ματιά, την άμεση ενημέρωση για το υπόλοιπο του πακέτου, χωρίς να απαιτείται κάποιος χειρισμός για να προσπελάσει τις Υπηρεσίες Ενημέρωσης Υπολοίπου. Η προτεινόμενη λύση μπορεί να λειτουργήσει ως συμπληρωματική των Υπηρεσιών Ενημέρωσης Υπολοίπου.

1.4 Τεχνικά χαρακτηριστικά – Απαιτήσεις.

Η εφαρμογή αναπτύχθηκε βάσει των παρακάτω Απαιτήσεων – Τεχνικών χαρακτηριστικών:

- Λειτουργικό σύστημα: Android
- Αρχιτεκτονική Εφαρμογής: Η εφαρμογή εκτελείται στο παρασκήνιο και επιτρέπει την συνεχή ένδειξη της κατανάλωσης του προπληρωμένου πακέτου. Δεν απαιτεί την

ύπαρξη εξειδικευμένης υποδομής (υλικού ή λογισμικού) στο δίκτυο, αλλά αξιοποιεί την υπάρχουσα υπηρεσία δωρεάν SMS του παρόχου, καθώς και την δυνατότητα του smartphone για τοπική παρακολούθηση της χρήσης.

- Μέθοδος πληρωμής υπηρεσιών: Προπληρωμένο πακέτο (prepay).
- Αριθμός δικτύων: Η εφαρμογή επεξεργάζεται στοιχεία ενός μόνο δικτύου.
- Υπηρεσίες: Αν και η αρχική απαίτηση αφορούσε στην ανάπτυξη εφαρμογής μόνον για την μέτρηση του υπολοίπου δεδομένων (data), περιελήφθηκαν στην εφαρμογή μετρήσεις υπολοίπου για φωνητικές κλήσεις και γραπτά μηνύματα.
- Μονάδες μέτρησης / προέλευση κίνησης:
    - Δεδομένα: megabyte, προς και από το ίδιο δίκτυο (on-net).
    - Φωνητικές κλήσεις: λεπτά ομιλίας προς το ίδιο δίκτυο (on-net).
    - SMS: γραπτά μηνύματα προς το ίδιο δίκτυο (on-net).

- Διεπαφή χρήστη: Προβλέπονται δύο όψεις, η Συνοπτική και η Αναλυτική.
    - Συνοπτική όψη: Ένα εικονίδιο ανά μετρούμενη υπηρεσία αντίστοιχα (δεδομένα, φωνητικές κλήσεις, γραπτά μηνύματα) με λευκό υπόβαθρο (background) στο εσωτερικό του. Το υπόβαθρο μειώνεται καθ'ύψος, με βήμα 10%, όσο το διαθέσιμο υπόλοιπο της υπηρεσίας εξαντλείται. Η ομάδα των τριών εικονιδίων εμφανίζεται στο πεδίο ειδοποιήσεων και επιπλέον είναι ορατή όποτε είναι ορατά τα εικονίδια στάθμης λαμβανομένου σήματος και βαθμού φόρτισης του συσσωρευτού.
    - Αναλυτική όψη: Για κάθε μετρούμενη υπηρεσία ο χρήστης έχει την δυνατότητα να δει αναλυτικά το υπόλοιπό της. Η εφαρμογή παρέχει δυο τέτοιες όψεις , την επέκταση του πεδίου ειδοποιήσεων και την κύρια όψη της εφαρμογής. Στην δεύτερη όψη περιλαμβάνει επίσης πλήκτρα καθορισμού μεγίστου ανά υπηρεσία (set max), επικαιροποίησης υπολοίπου (update) και βοήθειας (help), που οδηγούν στις ανάλογες όψεις.
- Επικαιροποίηση υπολοίπου (update): Η εφαρμογή ελέγχει σε τακτά χρονικά διαστήματα και επικαιροποιεί τους ενδείκτες ώστε να απεικονίζουν το σωστό υπόλοιπο των διαθέσιμων Megabytes, λεπτών ομιλίας και γραπτών μηνυμάτων του πελάτη.

Κεφάλαιο 2
Λειτουργικό σύστημα Android.

Στό Κεφάλαιο αυτό αναφέρονται γενικά στοιχεία και τεχνικά χαρακτηριστικά του λειτουργικού συστήματος Android, με το οποίο αναπτύχθηκε η εφαρμογή.

2.1 Λειτουργικό σύστημα Android.

Το Android είναι ένα λειτουργικό σύστημα, που αναπτύχθηκε από την Google, βασίζεται στον πυρήνα του Linux και έχει σχεδιαστεί κυρίως για φορητές συσκευές, όπως smartphones ή tablets με οθόνη αφής. Η διεπαφή χρήστη (user interface) βασίζεται κυρίως σε χειρονομίες αφής (όπως σύρσιμο, άγγιγμα) που επιδρούν επί των αντικειμένων της οθόνης και παράγουν αποτέλεσμα, όπως η ενεργοποίηση μιας εφαρμογής. Επιπλέον υπάρχει και ένα εικονικό πληκτρολόγιο για την εισαγωγή κειμένου, όταν απαιτείται. Εκτός από τις συσκευές με οθόνη αφής, η Google έχει αναπτύξει περαιτέρω το Android TV για τηλεοράσεις, το Android Auto για τα αυτοκίνητα και το Android Wear για ρολόγια χειρός, το καθένα με εξειδικευμένο περιβάλλον εργασίας χρήστη.

Το Android αναπτύχθηκε αρχικά από την εταιρία Android Inc, την οποία αγόρασε η Google το 2005. Η πρώτη έκδοση του Android παρουσιάσθηκε το 2007, μαζί με την ίδρυση του Open Handset Alliance, μιας κοινοπραξίας αρχικά 34 εταιριών λογισμικού, ηλεκτρονικών υλικών και τηλεπικοινωνιών, που αποσκοπούσε στην προώθηση ανοικτών προτύπων για φορητές συσκευές. Η Google διαθέτει τον κώδικα του Android με άδεια ελεύθερου λογισμικού, ωστόσο οι περισσότερες συσκευές Android περιλαμβάνουν και κλειστό (proprietary) λογισμικό, όπως π.χ το λογισμικό που απαιτείται για την πρόσβαση σε υπηρεσίες της Google.

Το Android είναι δημοφιλές σε εταιρείες που απαιτούν ένα έτοιμο, χαμηλού κόστους και προσαρμόσιμο λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας. Η ανοιχτή φύση του προσελκύει πολλούς προγραμματιστές, που χρησιμοποιούν τον κώδικά του ως βάση ανάπτυξης εφαρμογών. Τον Ιούλιο του 2013, το Google Play store διέθετε πάνω από ένα εκατομμύριο εφαρμογές Android, ενώ είχαν πραγματοποιηθεί πάνω από 50 δισεκατομμύρια λήψεις (downloads) αυτών των εφαρμογών. Στο ετήσιο συνέδριο Google I/O, τον Ιούνιο 2014, η εταιρεία αποκάλυψε ότι υπήρχαν πάνω από ένα δισεκατομμύριο ενεργοί μηνιαίοι

χρήστες Android, έναντι 538 εκατομμυρίων τον Ιούνιο του 2013. Το 2ο τετράμηνο του 2015 τα smartphone Android κυριαρχούν στην αγορά με μερίδιο 82,8%, έναντι 13,9% των iOS, 2,6% των Windows Phone, 0,3% των BlackBerry και 0,4% των υπολοίπων.

2.2 Χαρακτηριστικά του Android

2.2.1 Ανταλλαγή μηνυμάτων (Messaging).

Παρέχει τη δυνατότητα αποστολής και λήψης μηνυμάτων SMS και MMS, υπό μορφή νήματος (threaded) καθώς και επικοινωνία Android Cloud To Device Messaging (C2DM).

2.2.2 Φυλλομετρητής (Web browser)

Διαθέτει φυλλομετρητή που βασίζεται στον ανοιχτό κώδικα της μηχανής Blink σε συνδυασμό με την JavaScript Chrome V8. Υποστηρίζει HTML5 και CSS3.

2.2.3 Φωνητικές εντολές (Voice-based features)

Η αρχική έκδοση του Android είχε τη δυνατότητα φωνητικής αναζήτησης στο Google. Στην έκδοση Android 2.2 προστέθηκε η δυνατότητα κλήσεων, αποστολής γραπτών μηνυμάτων και πλοήγησης με φωνητικές εντολές. Στην έκδοση Android 4.1 έχει προστεθεί η δυνατότητα φωνητικής ανάγνωσης απαντήσεων του Knowledge Graph της Google.

2.2.4 Πολλαπλή επαφή (Multi-touch)

Το Android παρέχει εγγενή υποστήριξη εκτέλεσης εντολών που εισάγονται στην διεπαφή της οθόνης με ταυτόχρονη χρήση δύο δακτύλων. Ωστόσο, το χαρακτηριστικό αυτό ενεργοποιήθηκε με την κυκλοφορία του Nexus One.

2.2.5 Πολυδιεργασία (Multitasking)

Έχει δυνατότητα πολυδιεργασίας εφαρμογών, με πρόβλεψη ανάλογης κατανομής της μνήμης. Ο χρήστης μπορεί να μεταβαίνει από μια εφαρμογή σε άλλη και διάφορες εφαρμογές μπορούν να τρέχουν ταυτόχρονα.

2.2.6 Σύλληψη οθόνης (Screen capture)

Πριν από την έκδοση Android 4.0 οι μόνες μέθοδοι σύλληψης οθόνης ήταν μέσω κατάλληλων εφαρμογών ή με τη χρήση σύνδεσης PC. Μετά την έκδοση Android 4.0 προστέθηκε η δυνατότητα λήψης στιγμιότυπου οθόνης, πιέζοντας ταυτόχρονα συνδυασμό

πλήκτρων αναλόγως του τύπου της συσκευής.

2.2.7 Κλήσεις video (Video calling).

Το Android δεν υποστηρίζει εγγενώς κλήσεις βίντεο, αλλά ορισμένες συσκευές διαθέτουν ειδικές εκδόσεις λειτουργικού συστήματος που τις υποστηρίζει, είτε μέσω του δικτύου UMTS είτε μέσω IP (internet protocol). Από την έκδοση Android 2.3. παρέχεται η δυνατότητα βιντεοκλήσεων σε συνδυασμό με το Skype 2.1. Επίσης οι χρήστες της εφαρμογής Google+ για Android μπορούν να επικοινωνούν μεταξύ τους και με βιντεοκλήσεις.

2.2.8 Πολλαπλή υποστήριξη γλωσσών (Multiple language support)

Το Android υποστηρίζει πλήθος διαφορετικών γλωσσών, απλής και αμφίδρομης κατεύθυνσης γραφής κειμένου.

2.2.9 Προσιτότητα (Accessibility)

Για τα άτομα με μειωμένη ή καθόλου όραση παρέχεται η δυνατότητα εκφώνησης κειμένου (text-to-speech) μέσω της εφαρμογής TalkBack. Επίσης, για άτομα με προβλήματα ακοής υπάρχουν ανάλογα βοηθήματα.

2.2.10 Συνδεσιμότητα (Connectivity)

Το Android υποστηρίζει τεχνολογίες συνδεσιμότητας όπως GSM/EDGE, LTE, CDMA, EV-DO, UMTS, IDEN, Wi-Fi και WiMAX.

2.2.11 Άμεσο Wi-Fi (Wi-Fi Direct).

Πρόκειται για τεχνολογία που επιτρέπει στις εφαρμογές τον εντοπισμό και τη σύζευξη μεταξύ ομότιμων χρηστών (peer-to-peer), με σύνδεση μεγάλου εύρους ζώνης.

2.2.12 Bluetooth.

Υποστηρίζει φωνητική κλήση και αποστολή επαφών μεταξύ τηλεφώνων, αποστολή αρχείων (OPP), πρόσβαση στον τηλεφωνικό κατάλογο, διαχείριση ροών audio (A2DP) και έλεγχο από απόσταση συσκευών audio-video (AVRCP). Από την έκδοση Android 3.1 υποστηρίζονται επίσης πληκτρολόγιο, ποντίκι και joystick.

2.2.13 Πρόσδεση (Tethering).

Μετά την έκδοση Android 2.2 υποστηρίζεται η λειτουργία πρόσδεσης (tethering), η οποία επιτρέπει σε ένα τηλέφωνο να χρησιμοποιηθεί ως ασύρματο/ενσύρματο Wi-Fi hotspot.

2.2.14 Δέσμη Android (Android Beam).

Πρόκειται για εφαρμογή που βασίζεται στη τεχνολογία NFC (Near Field Communication), η οποία επιτρέπει την άμεση αποκατάσταση επικοινωνίας μεταξύ δύο συσκευών, όταν πλησιάσουν πάρα πολύ.

2.2.15 Υποστήριξη ροής πολυμέσων (Streaming media support).

Υποστηρίζει τα παρακάτω πρότυπα εικόνας/ήχου/ video:
H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF και BMP

2.2.16 Εσωτερική αποθήκευση (Storage).

Χρησιμοποιείται η SQLite, μια 'ελαφριά΄ βάση δεδομένων.

2.2.17 Εξωτερική αποθήκευση (External storage).

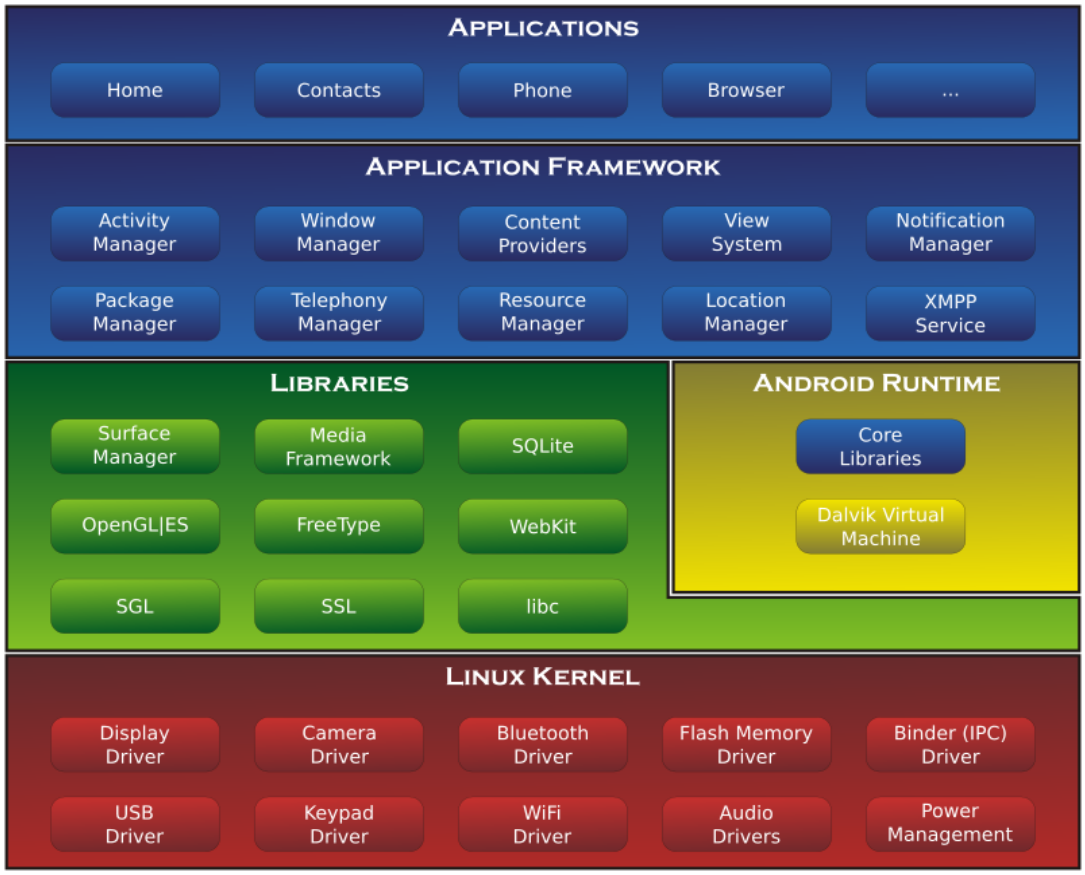
Οι περισσότερες συσκευές Android περιλαμβάνουν υποδοχή κάρτας microSD, που μπορεί να διαβάσει κάρτες διαμορφωμένες σύμφωνα με το σύστημα αρχείων FAT32, ext3 ή ext4. Για να επιτραπεί η χρήση μέσων αποθήκευσης υψηλής χωρητικότητας, όπως USB flash drives και σκληρών δίσκων USB, απαιτείται καλώδιο τύπου USB-OTG. Η διαχείριση αρχείων μορφοποιημένων κατά FAT32 γίνεται μέσω του οδηγού VFAT του πυρήνα Linux, ενώ για τα αρχεία τύπου NTFS, HFS Plus και exFAT απαιτούνται οδηγοί που παρέχονται από ανεξάρτητους κατασκευαστές .

2.3 Αρχιτεκτονική

Το Λειτουργικό Σύστημα Android είναι μια στοίβα στοιχείων λογισμικού, που χονδρικά χωρίζεται σε πέντε τμήματα και τέσσερις κύρια στρώματα, όπως φαίνεται παρακάτω στην Εικόνα 2.1.

2.3.1 Πυρήνας Linux (Linux Kernel)

Ο πυρήνας Linux βρίσκεται στο κάτω μέρος της στοίβας λογισμικού και παρέχει ένα επίπεδο αφαίρεσης μεταξύ του υλικού της συσκευής και τα ανώτερα στρώματα του



Εικόνα 2.1. Διάγραμμα Αρχιτεκτονικής Android

λογισμικού Android. Βασισμένος στην έκδοση 3.6 του Linux, ο πυρήνας εξασφαλίζει προληπτική πολυδιεργασία (preemptive multitasking), χαμηλού επιπέδου υπηρεσίες πυρήνα του συστήματος, όπως διαχείριση μνήμης και ενέργειας. Επιπλέον στο στρώμα αυτό περιλαμβάνονται τα προγράμματα οδήγησης υλικού, όπως η οθόνη της συσκευής, το Wi-Fi, ο ήχος κ.λ.π.

2.3.2 Android runtime

Το τμήμα αυτό περιέχει ένα βασικό συστατικό που ονομάζεται Dalvik Virtual Machine το οποίο είναι ένα είδος Java Virtual Machine, ειδικά σχεδιασμένο και βελτιστοποιημένο για κινητές συσκευές, που λειτουργούν με συσσωρευτή και διαθέτουν περιορισμένες δυνατότητες σε μνήμη και CPU. Η Dalvik VM κάνει χρήση των βασικών χαρακτηριστικών του Linux, όπως η διαχείριση της μνήμης και multi-threading. Η Dalvik VM ενεργοποιεί την κάθε εφαρμογή Android ώστε να τρέξει τη δική της διαδικασία. Το Android runtime παρέχει επίσης ένα σύνολο βασικών βιβλιοθηκών (Core Libraries) που επιτρέπουν σε προγραμματιστές να γράψουν εφαρμογές Android, χρησιμοποιώντας την γλώσσα Java.

2.3.3 Βιβλιοθήκες Android (Android Libraries).

Αυτή η κατηγορία περιλαμβάνει τις βιβλιοθήκες που βασίζονται σε Java, ειδικά για ανάπτυξη του Android. Παραδείγματα βιβλιοθηκών σε αυτή την κατηγορία είναι οι βιβλιοθήκες ανάπτυξης της διεπαφής χρήστη, της σχεδίασης γραφικών και της πρόσβασης σε βάσεις δεδομένων.

2.3.4 Πλαίσιο εφαρμογής (Application Framework).

Το πλαίσιο εφαρμογής είναι ένα σύνολο υπηρεσιών που αποτελούν το περιβάλλον στο οποίο οι εφαρμογές Android 'τρέχουν' και διαχειρίζονται. Οι υπηρεσίες αυτές είναι ενδεικτικά οι παρακάτω:

- **Διαχείριση Δραστηριότητας** (Activity Manager): Παρακολουθεί και ελέγχει κάθε λειτουργία της εφαρμογής.
- **Πάροχοι Περιεχομένου** (Content Providers): Επιτρέπει σε εφαρμογές να δημοσιοποιούν και να ανταλλάσσουν δεδομένα με άλλες εφαρμογές.
- **Διαχείριση Ειδοποιήσεων** (Notifications Manager): Επιτρέπει σε εφαρμογές να εμφανίζουν ειδοποιήσεις στο χρήστη.
- **Διαχείριση Πακέτου** (Package Manager): Επιτρέπει στις εφαρμογές να αντλούν πληροφορίες σχετικά με άλλες εφαρμογές, που είναι εγκατεστημένες στη συσκευή.
- **Διαχείριση Τηλεφωνίας** (Telephony Manager): Παρέχει πληροφορίες στην εφαρμογή για τις διαθέσιμες υπηρεσίες τηλεφωνίας, όπως και για την κατάσταση του συνδρομητή.
- **Διαχείριση Τοποθεσίας** (Location Manager): Παρέχει πρόσβαση στις υπηρεσίες εντοπισμού και επιτρέπει στην εφαρμογή να λαμβάνει ενημερώσεις σχετικά με τις αλλαγές θέσης.

2.3.5 Εφαρμογές (Applications).

Στην κορυφή της στοίβας λογισμικού Android είναι οι εφαρμογές. Σε αυτές περιλαμβάνονται τόσο οι εγγενείς εφαρμογές Android (π.χ. web browser και e-mail), που είναι εγκατεστημένες εξ΄αρχής στη συσκευή, αλλά και οι εφαρμογές τρίτων που έχουν εγκατασταθεί από το χρήστη, μετά την αγορά της.

Κεφάλαιο 3
Επικοινωνία Ανθρώπου - Υπολογιστή.

Θεμελιώδης απαίτηση για την εφαρμογή που αναπτύχθηκε ήταν να παρέχει στο χρήστη την δυνατότητα εύκολου χειρισμού. Στο Κεφάλαιο αυτό γίνεται σύντομη αναφορά στην έννοια της Αλληλεπίδρασης Ανθρώπου - Υπολογιστή και στις Αρχές Σχεδιασμού Διεπαφών Οθόνης. Επιπλέον, μετά την ανάλυση κάθε μιας των Αρχών Σχεδιασμού, περιγράφεται το πως η υπόψη Αρχή υλοποιήθηκε στην εφαρμογή Trindicator.

3.1 Αλληλεπίδραση Ανθρώπου-Υπολογιστή (ΑΑΥ)

Η Αλληλεπίδραση Ανθρώπου -Υπολογιστή (ΑΑΥ) ερευνά τη σχεδίαση και τη χρήση της τεχνολογίας των υπολογιστών, με ιδιαίτερη έμφαση στη διεπαφή μεταξύ ανθρώπων και υπολογιστών. Οι ερευνητές στον τομέα της ΑΑΥ παρατηρούν τους τρόπους με τους οποίους οι άνθρωποι αλληλεπιδρούν με τους υπολογιστές και σχεδιάζουν τεχνολογίες που επιτρέπουν στους ανθρώπους να αλληλεπιδρούν με τους υπολογιστές με νέους τρόπους. Ως πεδίο έρευνας η ΑΑΥ αντλεί στοιχεία από την επιστήμη των υπολογιστών, τις επιστήμες συμπεριφοράς, σχεδιασμού, μελέτης των μέσων ενημέρωσης και πολλούς άλλους επιστημονικούς τομείς.

Οι άνθρωποι αλληλεπιδρούν με τους υπολογιστές με πολλούς τρόπους και η διεπαφή που χρησιμοποιείται μεταξύ ανθρώπων και υπολογιστών είναι ζωτικής σημασίας για τη διευκόλυνση αυτής της αλληλεπίδρασης. Τα παραδείγματα αλληλεπίδρασης είναι πολλά και εκτείνονται από την απλή καθημερινή χρήση εφαρμογών στον προσωπικό υπολογιστή ή τον υπολογιστή χειρός μέχρι την υποβοήθηση χειρισμού πλοίων και αεροσκαφών.

Κακοσχεδιασμένες διεπαφές ανθρώπου-υπολογιστή μπορεί να οδηγήσουν σε απρόβλεπτα προβλήματα. Ένα κλασικό παράδειγμα είναι το πυρηνικό ατύχημα στο εργοστάσιο παραγωγής ενέργειας Three Mile Island, όπου οι έρευνες κατέληξαν στο συμπέρασμα ότι ο σχεδιασμός της διεπαφής ανθρώπου-υπολογιστή ήταν τουλάχιστον εν μέρει υπεύθυνος για την καταστροφή. Ομοίως, έχουν σημειωθεί αεροπορικά ατυχήματα επειδή οι κατασκευαστές υιοθέτησαν μη τυποποιημένο σχεδιασμό οργάνων πτήσης. Αν και τα νέα όργανα υπερτερούσαν από απόψεως χρηστικότητας, οι πιλότοι είχαν ήδη ριζωμένη την "πρότυπη"

διάταξη και η σχεδιαστικά καλύτερη ιδέα είχε ανεπιθύμητα αποτελέσματα στην πράξη.

3.2 Οι Αρχές σχεδιασμού διεπαφών οθόνης.

Ο Christopher Wickens et al. καθορίζει 13 αρχές για τον σχεδιασμό διεπαφών οθόνης στο βιβλίο An Introduction to Human Factors Engineering. Οι αρχές αυτές διέπουν την ανθρώπινη αντίληψη και την επεξεργασία πληροφοριών και μπορούν να χρησιμοποιηθούν για τον αποτελεσματικό σχεδιασμό οθόνης. Η εφαρμογή αυτών των αρχών εξασφαλίζει αφενός μεν την μείωση των σφαλμάτων χειρισμού και του απαιτούμενου χρόνου εκπαίδευσης , αφετέρου δε την αύξηση της αποτελεσματικότητα και της ικανοποίησης των χρηστών.

Ορισμένες αρχές ενδέχεται να μην μπορούν να εφαρμοσθούν σε κάποιους σχεδιασμούς οθονών. Μερικές αρχές μπορεί να φαίνονται αντικρουόμενες και κάποιες φορές είναι δύσκολο να αποφασισθεί ότι μια αρχή είναι σημαντικότερη άλλης. Η επίτευξη λειτουργικής ισορροπίας μεταξύ των αρχών είναι ζωτικής σημασίας για τον αποτελεσματικό σχεδιασμό.

3.2.1 Αρχές αντίληψης του χρήστη.

Οι αρχές που σχετίζονται με την αντίληψη του χρήστη εξετάζουν τον τρόπο που ο χρήστης αντιλαμβάνεται αρχικά το υλικό που του διατίθεται. Προκειμένου να αποφευχθεί η σύγχυση του χρήστη, οι πληροφορίες πρέπει να παρουσιάζονται με σαφή και μη διφορούμενο τρόπο. Υπάρχουν πέντε Αρχές Αντίληψης του χρήστη που αναφέρονται παρακάτω:

3.2.1.1 Ευανάγνωστες οθόνες.

Η αναγνωσιμότητα της οθόνης είναι ο πλέον κρίσιμος παράγοντας σχεδιασμού της διεπαφής. Κάθε οθόνη πρέπει να είναι ευανάγνωστη ώστε να επιτρέπει στον χρήστη να αλληλεπιδράσει με επιτυχία. Πρέπει να χρησιμοποιείται ο σωστός συνδυασμός χρωμάτων, αντιθέσεων και ήχων για να εξασφαλιστεί ότι ο χρήστης αντλεί τις απαραίτητες πληροφορίες από την οθόνη. Επιπλέον για διεπαφές οθονών αφής πρέπει να εξασφαλίζεται και η απρόσκοπτη δυνατότητα επιλογής – εισαγωγής στοιχείων.

Τρόπος Υλοποίησης της Αρχής: Σε όλες τις οθόνες της διεπαφής που υλοποιήθηκε έχουν χρησιμοποιηθεί ενδείκτες και πλήκτρα ικανού μεγέθους, με επεξηγηματικό κείμενο για την παρεχόμενη λειτουργία. Το μέγεθος των πλήκτων επιλέχθηκε έτσι ώστε να επιτρέπει τον χειρισμό τους από χειριστή με φυσιολογικό μέγεθος δακτύλων.

3.2.1.2 Αποφυγή εκτίμησης μεταβλητών βάσει υποκειμενικών κριτηρίων.

Η άντληση πληροφοριών από το χρήστη δεν πρέπει να βασίζεται σε εκτίμηση του μεγέθους

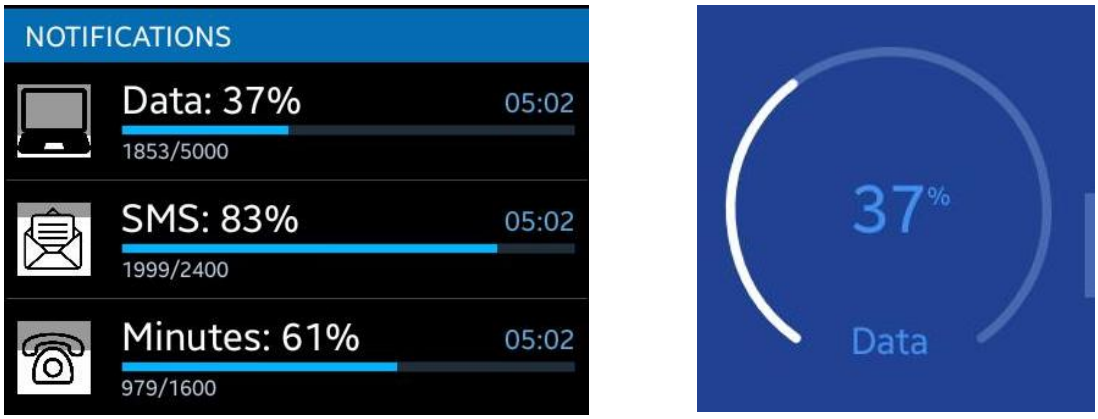
μιας συνεχούς μεταβλητής, όπως χρώμα, ένταση κλπ, επειδή οι μεταβλητές αυτές μπορεί να έχουν πολλές δυνατές τιμές. Ο χρήστης, λόγω εσφαλμένης κρίσης, πιθανόν να εκτιμήσει λανθασμένα το προβαλλόμενο μέγεθος. Παράδειγμα αποτελεί ένα φανάρι δύο φωτεινών καταστάσεων, που εναλλάσσει τα χρώματα κίτρινο και πορτοκαλί, όπου κάθε χρώμα αντιστοιχεί σε μια ενέργεια και ισχύουν οι παρακάτω οδηγίες:

Αν το φανάρι είναι κίτρινο εκτελέστε την ενέργεια Α.
Αν το φανάρι είναι πορτοκαλί εκτελέστε την ενέργεια Β.

Εικόνα 3.1.

Γίνεται άμεσα αντιληπτό ότι ο χρήστης θα δυσκολευθεί και θα καθυστερήσει την εκτέλεση των ενεργειών, μέχρι να αποφασίσει αν το χρώμα είναι πορτοκαλί ή κίτρινο.

Τρόπος Υλοποίησης της Αρχής: Τα υπόλοιπα χρήσης του χρόνου ομιλίας, των δεδομένων και των συντόμων μηνυμάτων απεικονίζονται πλην του τοξοειδούς ή γραμμικού ενδείκτη και με αριθμητική τιμή , έτσι ώστε ο χρήστης δεν απαιτείται να εκτιμήσει το μέγεθος του τόξου ή του ευθύγραμμου τμήματος, αλλά απλά διαβάζει τον προβαλλόμενο αριθμό, όπως φαίνεται στην παρακάτω Εικόνα 3.2



Εικόνα 3.2. Ενδείκτες της εφαρμογής.

3.2.1.3 Επεξεργασία με βάση την εμπειρία και την προσδοκία.

Οι άνθρωποι αντιλαμβάνονται και ερμηνεύουν τις πληροφορίες με βάση την προηγούμενη εμπειρία και το τι προσδοκούν να δουν. Εάν μια οθόνη περιέχει πληροφορίες που διαφέρει από τις προσδοκίες τους μπορεί να δουν αυτό που ανέμεναν, αντί αυτού που προβάλλεται. Για τον λόγο αυτό πληροφορίες που είναι αντίθετες των προσδοκιών πρέπει να παρουσιάζονται με τρόπο που να εφιστά ιδιαίτερη προσοχή στον διαφοροποιούμενο τομέα. Ένα παράδειγμα επεξεργασίας με βάση την εμπειρία και την προσδοκία φαίνεται στην Εικόνα3.3. Ακόμη και αν το δεύτερο γράμμα σε κάθε λέξη είναι διφορούμενο, η επεξεργασία

με βάση την εμπειρία και την προσδοκία επιτρέπει την εύκολη αποσαφήνιση τις φράσης, από τα συμφραζόμενα.

Εικόνα 3.3.

Τρόπος Υλοποίησης της Αρχής: Για την αντιστοίχιση των ενδείξεων με τις μετρούμενες υπηρεσίες έχουν επιλεγεί κατάλληλα εικονίδια, που παραπέμπουν στην ανάλογη υπηρεσία. Συγκεκριμένα για τις κλήσεις ομιλίας έχει επιλεγεί εικονίδιο τηλεφώνου, για τα σύντομα μηνύματα εικονίδιο φακέλλου επιστολής και για τα δεδομένα εικονίδιο ηλεκτρονικού υπολογιστή. Εκτιμάται ότι ο χρήστης πολύ εύκολα θα συσχετίσει τις μετρούμενες υπηρεσίες με τα υπόψη εικονίδια, που φαίνονται στην Εικόνα 3.4 .

Εικόνα 3.4. Εικονίδια σήμανσης για κλήσεις ομιλίας, σύντομα μηνύματα και δεδομένα.

(Icons made by Freepik from www.flaticon.com)

3.2.1.4 Το κέρδος του πλεονασμού.

Εάν μια πληροφορία παρουσιάζεται περισσότερο από μια φορά, είναι πιθανό ότι θα γίνει ευκολότερα κατανοητή. Αυτό μπορεί να επιτευχθεί με την παρουσίαση της πληροφορίας σε εναλλακτικές φυσικές μορφές (π.χ. χρώμα και σχήμα, φωνή και εκτύπωση, κλπ.), καθώς ο πλεονασμός δεν είναι πάντοτε περιττός. Ένα παράδειγμα κέρδους λόγω πλεονασμού αποτελεί το φανάρι ρύθμισης κυκλοφορίας, όπου συνδυάζονται το χρώμα και η θέση του αναμμένου λαμπτήρα για την αποκωδικοποίηση του προβαλλόμενου μηνύματος.

Τρόπος Υλοποίησης της Αρχής: Τα υπόλοιπα χρήσης του χρόνου ομιλίας, των δεδομένων και των συντόμων μηνυμάτων απεικονίζονται πλην του τοξοειδούς ή γραμμικού ενδείκτη και αριθμητικά, όπως φαίνεται και στην Εικόνα 3.2. Συνδυάζεται έτσι το πλεονέκτημα της αναλογικής απεικόνισης (ανάγνωση μετρουμένου μεγέθους με μια φευγαλέα ματιά) με εκείνο της ψηφιακής απεικόνισης (ακρίβεια).

3.2.1.5 Αποφυγή ομοιότητας.

Πληροφορίες που είναι παρόμοιες είναι πιθανό να προκαλέσουν σύγχυση. Για παράδειγμα το A423B9 και το A423B8 μπορούν να εκληφθούν ως ίδια. Τα περιττά κοινά χαρακτηριστικά πρέπει να αφαιρούνται, ενώ πρέπει να τονίζονται τα διαφορετικά

χαρακτηριστικά.

Τρόπος Υλοποίησης της Αρχής: Καταβλήθηκε προσπάθεια να αποφευχθεί οποιαδήποτε ομοιότητα στην απεικόνιση των μετρουμένων μεγεθών, με χρήση εικονιδίων και κειμένου που καθιστούν εμφανή την μεταξύ τους διαφοροποίηση. Επιπλέον τα εικονίδια που επιλέχθηκαν διαφέρουν σημαντικά εκείνων που ήταν προεγκατεστημένα στην συσκευή.

3.2.2 Αρχές του διανοητικού μοντέλου του χρήστη.

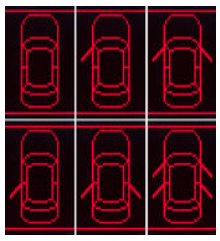
Όταν ο χρήστης βλέπει μια οθόνη η ερμηνεία της βασίζεται συνήθως στις προσδοκίες του για το σύστημα αυτό. Οι προσδοκίες αυτές προέρχονται από τις εμπειρίες του παρελθόντος , που έχουν σχηματίσει ένα νοητικό μοντέλο του συστήματος και της λειτουργίας του. Είναι σημαντικό να σχεδιάζονται οθόνες συνεπείς με το νοητικό μοντέλο του χρήστη. Υπάρχουν δύο Αρχές Διανοητικού Μοντέλου που αναφέρονται παρακάτω:

3.2.2.1 Εικαστικός ρεαλισμός.

Μια οθόνη πρέπει να ταιριάζει οπτικά και λειτουργικά με τη μεταβλητή που αντιπροσωπεύει. Ισχύει ο γενικός κανόνας ότι οι συνεχείς μεταβλητές απεικονίζονται αναλογικά, ενώ οι διακριτές μεταβλητές ψηφιακά. Επίσης οι υψηλές τιμές των μεταβλητών πρέπει να είναι στην κορυφή της οθόνης (ή δεξιά) και οι χαμηλές τιμές στο κάτω μέρος (ή αριστερά), όπως στην Εικόνα 3.5.

Εικόνα 3.5. Αναλογικό θερμόμετρο.

Επιπλέον, αν υπάρχουν πολλαπλά στοιχεία για απεικόνιση πρέπει να μορφοποιηθούν έτσι ώστε να αντιστοιχούν στο περιβάλλον που αντιπροσωπεύουν. Μια εφαρμογή αυτής της αρχής φαίνεται στην οθόνη ελέγχου θυρών, της Εικόνας 3.6, όπου απεικονίζεται όχι μόνον ότι κάποια πόρτα είναι ανοικτή, αλλά και ποιά ή ποιές συγκεκριμένα.

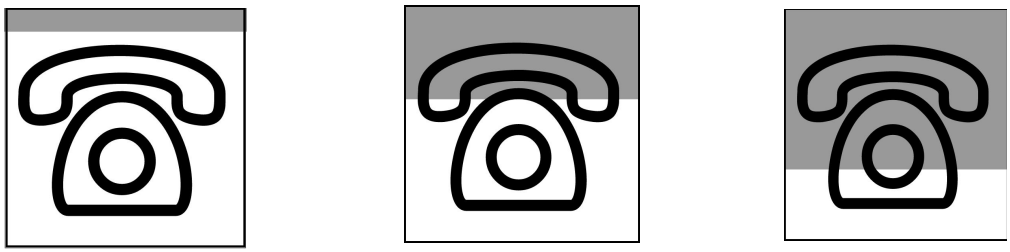

Εικόνα 3.6. Οθόνη ελέγχου θυρών.

Τρόπος Υλοποίησης της Αρχής: Τηρείται ο κανόνας που επιβάλλει οι υψηλές τιμές των μεταβλητών να απεικονίζονται στο δεξί τμήμα των ενδεικτών, όπως στην Εικόνα 3.2.

3.2.2.2 Αρχή του κινούμενου μέρους.

Το ίχνος και η κατεύθυνση κίνησης των απεικονιζομένων μεταβλητών θα πρέπει να είναι συμβατά με το νοητικό μοντέλο του χρήστη, όπως αυτό έχει διαμορφωθεί από τη φυσική παρατήρηση. Για παράδειγμα, το κινούμενο στοιχείο σε ένα μετρητή υψομέτρου θα πρέπει να κινείται προς τα επάνω, όταν το υψόμετρο αυξάνεται.

Τρόπος Υλοποίησης της Αρχής: Σε κάθε ένα εικονίδιο ανά μετρούμενη υπηρεσία (δεδομένα, φωνητικές κλήσεις, γραπτά μηνύματα) υπάρχει λευκό υπόβαθρο (background) στο εσωτερικό του. Το υπόβαθρο μειώνεται καθ'ύψος, με βήμα 10%, όσο το διαθέσιμο υπόλοιπο της υπηρεσίας εξαντλείται, όπως στην Εικόνα 3.10.



Εικόνα 3.10. Διαθέσιμο υπόλοιπο κλήσεων ομιλίας 90, 60, 30% του αρχικού.
(Icons made by Freepik from www.flaticon.com)

3.2.3 Αρχές με βάση την προσοχή του χρήστη.

Εάν η διεπαφή περιέχει πολλά στοιχεία ο χρήστης πρέπει να κατανέμει την προσοχή του μεταξύ των στοιχείων της, ενώ περιστασιακά θα απαιτηθεί να εστιάσει την προσοχή του σε περισσότερα από ένα στοιχεία ταυτόχρονα. Για τον λόγο αυτό η διεπαφή πρέπει να είναι σχεδιασμένη με την κατάλληλη διάταξη, που να επιτρέπει στο χρήστη να έχει εύκολη πρόσβαση στις απαραίτητες πληροφορίες. Υπάρχουν τρεις Αρχές με βάση την προσοχή του χρήστη , όπως παρακάτω:

3.2.3.1 Ελαχιστοποίηση του κόστους πρόσβασης στις πληροφορίες.

Όταν η προσοχή του χρήστη εκτρέπεται από το ένα σημείο στο άλλο για να έχει πρόσβαση στις απαραίτητες πληροφορίες, υπάρχει ένα σχετικό κόστος, σε χρόνο και κόπο. Η σχεδίαση οθόνης θα πρέπει να ελαχιστοποιεί αυτό το κόστος, προβλέποντας τα σημεία παροχής της πληροφορίας να είναι όσο το δυνατόν πλησιέστερα μεταξύ τους, χωρίς ωστόσο να παραβιάζεται η αρχή της επαρκούς αναγνωσιμότητας, που προαναφέρθηκε.

Τρόπος Υλοποίησης της Αρχής: Όπως φαίνεται στην Εικόνα 3.11 παρέχεται η δυνατότητα

στο χρήστη να πληροφορηθεί άμεσα το υπόλοιπο ανά υπηρεσία, παρατηρώντας τα εικονίδια που προβάλλονται συνεχώς στο πάνω αριστερό τμήμα της οθόνης (μπάρα ειδοποιήσεων), χωρίς να χρειασθεί να καλέσει την ίδια την εφαρμογή Trindicator.

Εικόνα 3.11. Άμεση πληροφόρηση υπολοίπου

3.2.3.2 Συμβατότητα εγγύτητας.

Η κατανομή της προσοχής μεταξύ δύο ή περισσοτέρων πηγών πληροφοριών μπορεί να είναι απαραίτητη για την ολοκλήρωση ενός έργου. Σε αυτή την περίπτωση επιβάλλεται να είναι μικρή η προσπάθεια πρόσβασης στις πληροφορίες, και αυτό επιτυγχάνεται με πολλούς τρόπους (π.χ. εγγύτητα, σύνδεση με κοινά χρώματα, σχήματα, κλπ). Ειδικά για τις οθόνες των smartphones, που έχουν περιορισμένες διαστάσεις, η πυκνή απεικόνιση μπορεί να προκαλέσει δυσκολία άντλησης πληροφοριών και λανθασμένους χειρισμούς.

Τρόπος Υλοποίησης της Αρχής: Η διάταξη των εικονιδίων, των ενδεικτών και των πλήκτρων της διεπαφής εξασφαλίζει την απρόσκοπτη άντληση των πληροφοριών και τον εύκολο χειρισμό. Αυτό αναλύεται διεξοδικότερα στο επόμενο Κεφάλαιο 4, όπου παρουσιάζονται και επεξηγούνται οι επιμέρους όψεις της εφαρμογής Trindicator.

3.2.3.3 Πολλαπλοί πόροι.

Ο χρήστης μπορεί να επεξεργαστεί ευκολότερα πληροφορίες από διαφορετικές πηγές. Για παράδειγμα, μπορούν να παρουσιαστούν ταυτόχρονα οπτικές και ακουστικές πληροφορίες αντί να παρουσιασθούν πρώτα όλες οι οπτικές ή όλες οι ακουστικές πληροφορίες. (Δεν υπήρξε απαίτηση που να ικανοποιείται με τη χρήση αυτής της αρχής)

3.2.4 Αρχές με βάση την μνήμη του χρήστη.

Ανθρώπινη μνήμη έχει περιορισμούς και κάνει λάθη πολύ εύκολα. Στη βραχυπρόθεσμη μνήμη διατηρείται μόνο μια μικρή ποσότητα πληροφοριών, ενώ οι πληροφορίες της μακροπρόθεσμης μνήμης λησμονούνται με την πάροδο του χρόνου. Οι αρχές που σχετίζονται με την μνήμη του χρήστη αντιμετωπίσουν αυτά τα ζητήματα και παρέχουν

μεθόδους αντιμετώπισης. Υπάρχουν τρεις Αρχές για την μνήμη του χρήστη, όπως παρακάτω:

3.2.4.1 Υποκατάσταση μνήμης από οπτικές ή ηχητικές πληροφορίες.

Ο χρήστης δεν πρέπει να χρειάζεται να διατηρεί σημαντικές ή πολλές γνώσεις στη μνήμη του, παρά μόνο τις απολύτως αναγκαίες για την λειτουργία της διεπαφής. Οι λοιπές πληροφορίες πρέπει να παρέχονται υπό μορφή βοηθημάτων (μενού βοήθειας, λίστα ελέγχου κ.λπ) ώστε να αποφορτίζεται η μνήμη του, χωρίς να προκαλείται υπερφόρτωση πληροφοριών. Σε ένα αποτελεσματικό σχεδιασμό η γνώση στη μνήμη του χρήστη και η γνώση υπό μορφή βοηθημάτων πρέπει να είναι ισορροπημένες και το γεγονός αυτό αποτελεί πρόκληση για τον σχεδιαστή.

Τρόπος Υλοποίησης της Αρχής: Για την υποστήριξη του χρήστη παρέχεται επιλογή βοήθειας, που καλείται μέσω της εφαρμογής και περιγράφει περιληπτικά τις επιμέρους λειτουργίες της.

3.2.4.2 Αρχή της συμβατότητας.

Κατά την ανάπτυξη νέων διεπαφών είναι σκόπιμο να εξετάζεται κατά πόσο είναι δυνατό να χρησιμοποιηθούν, τμηματικά η συνολικά, παλαιότερες επιτυχημένες και ευρείας αποδοχής σχεδιάσεις. Με τον τρόπο αυτό παρέχεται η δυνατότητα σε μεγάλο αριθμό χρηστών να χειρισθούν με ευκολία την διεπαφή, καθώς είναι εξοικειωμένοι με τα εικονίδια, τις ενέργειες και διαδικασίες, εφόσον έχουν σχεδιασθεί κατά τρόπο συμβατό ως προς προγενέστερες σχεδιάσεις. Τα προγράμματα της Microsoft είναι τυπικό παράδειγμα αυτής της αρχής, καθώς τα περισσότερα από τα προϊόντα της μοιράζονται ένα τυποποιημένο κύριο μενού λειτουργίας.

Τρόπος Υλοποίησης της Αρχής: Η όλη λειτουργικότητα της εφαρμογής Trindicator είναι συμβατή με τα πρότυπα απεικόνισης και χειρισμού που έχει καθιερώσει το λειτουργικό Android. Ένα παράδειγμα αυτής της συμβατότητας αποτελούν τα εικονίδια πληροφόρησης για το υπόλοιπο ανά υπηρεσία (φωνητικές κλήσεις, γραπτά μηνύματα, δεδομένα), τα οποία εμφανίζονται στο πεδίο ειδοποιήσεων και οι ενδείξεις τους μεταβάλλονται κατά τον ίδιο τρόπο που μεταβάλλονται οι ενδείξεις ισχύος του λαμβανόμενου σήματος και του συσσωρευτού της συσκευής.

3.2.4.3 Προβλεπτική (έξυπνη) υποβοήθηση (Predictive aiding).

Μια διεπαφή θα πρέπει να δείχνει την μελλοντική κατάσταση, αντί να απαιτεί από τους χρήστες την εκτίμηση των επιμέρους παραγόντων που σχετίζονται με την κατάσταση αυτή,

προκειμένου να λάβουν μια απόφαση. Έτσι εξαλείφονται οι πολύπλοκες νοητικές εργασίες και αντικαθίστανται με άλλες πιο απλές, για τη μείωση του φόρτου του χρήστη. Αυτό επιτρέπει στο χρήστη να εστιάσει όχι μόνο στις τρέχουσες συνθήκες, αλλά και στις μελλοντικές. Παράδειγμα προβλεπτικής υποβοήθησης αποτελεί η ένδειξη της μπαταρίας φορητής συσκευής, όπου εκτός από την κατάσταση φόρτισης απεικονίζεται και το εκτιμώμενο υπόλοιπο χρόνου λειτουργίας της συσκευής. Απαλλάσσεται έτσι ο χρήστης από τον νοητικό φόρτο να αναγάγει την κατάσταση φόρτισης σε χρόνο λειτουργίας της συσκευής, βάσει της προσωπικής του εμπειρίας. Ωστόσο, κρίνεται σκόπιμο να επισημανθεί ότι η προβλεπτική υποβοήθηση λειτουργεί με ακρίβεια όταν οι παράγοντες επί των οποίων βασίζεται παραμένουν σχετικώς αμετάβλητοι κατά το διάστημα παρατήρησης.

Τρόπος Υλοποίησης της Αρχής: Όταν το υπόλοιπο μιας εκ των τριών υπηρεσιών είναι μικρότερο του 10% εμφανίζεται στο αντίστοιχο εικονίδιο το σύμβολο " ! ", με ερυθρό χρώμα και ο χρήστης ενημερώνεται άμεσα ότι επίκειται η εκπνοή της προπληρωμένης υπηρεσίας. Στην Εικόνα 3.12 απεικονίζεται η επικείμενη εκπνοή του χρόνου φωνητικών κλήσεων.

Εικόνα 3.12. Εικονίδιο τέλος χρόνου
(Icon made by Freepik from www.flaticon.com)

Κεφάλαιο 4

Οι Όψεις της Διεπαφής.

Στο Κεφάλαιο αυτό παρουσιάζονται οι όψεις της διεπαφής της εφαρμογής Trindicator και ο τρόπος μετάβασης από την μια όψη στην άλλη.

4.1 Συντόμευση (Shortcut)

Η πρώτη επαφή που έχει ο χρήστης με την εφαρμογή είναι η συντόμευση Trindicator, που εμφανίζεται στο μενού της συσκευής και επιτρέπει την εκκίνηση της εφαρμογής, όπως φαίνεται στην Εικόνα 4.1.

Εικόνα 4.1. Συντόμευση εφαρμογής Trindicator.

4.2 Ειδοποιήσεις (Notifications).

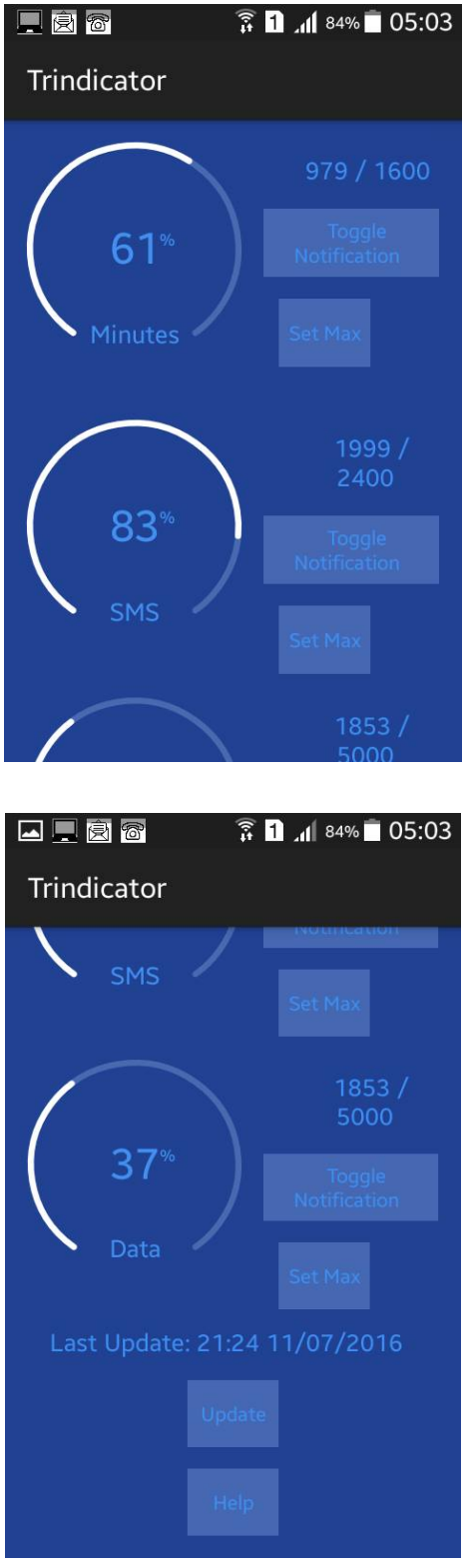
Ίσως η πλέον σημαντική όψη της εφαρμογής και αυτή η οποία ο χρήστης θα συμβουλεύεται περισσότερο. Η εφαρμογή επιτρέπει στον χρήστη να προβάλει ειδοποιήσεις για κανένα, ένα, δύο ή όλα τα μετρούμενα μεγέθη. Ανάλογα με το ποιό μέγεθος συμβολίζουν και το υπόλοιπό τους (εκφρασμένο σαν ποσοστό) οι ειδοποιήσεις έχουν το αντίστοιχο εικονίδιο. Το εικονίδιο αυτό αποτελεί μια συμβολική έκφραση του μετρούμενου μεγέθους που "αδειάζει" καθώς το υπόλοιπο μειώνεται, κατ αναλογία του αντίστοιχου εικονιδίου της μπαταρίας της συσκευής. Επεκτείνοντας το πεδίο των ειδοποιήσεων ο χρήστης βλέπει σε πιο αναλυτική μορφή πληροφορίες σχετικά με τα εν λόγω μεγέθη. Προβάλλονται εκεί τόσο το υπόλοιπο και η τρέχουσα τιμή αναφοράς όσο και το ποσοστό. Επιπλέον το υπόλοιπο εμφανίζεται και σαν ραβδοειδής απεικόνιση, που "γεμίζει" ή "αδειάζει". Τέλος με πάτημα

Εικόνα 4.2. Όψη Ειδοποιήσεων.

πάνω στην ειδοποίηση ο χρήστης μπορεί να μεταβεί στην κεντρική όψη της εφαρμογής.

4.3 Κεντρική όψη.

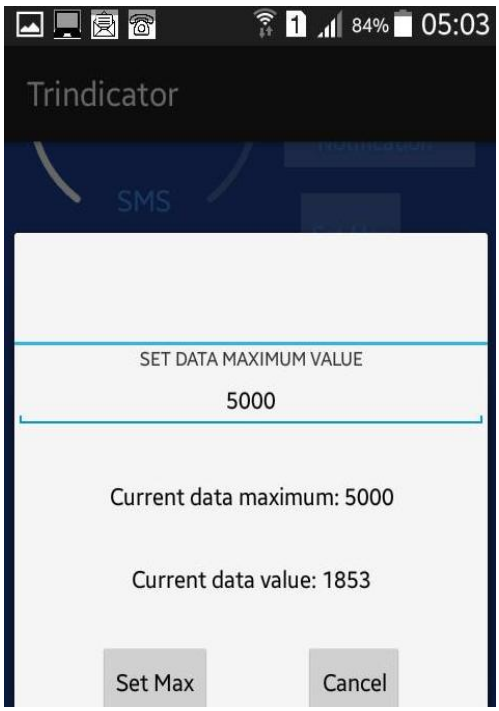
Η βασική όψη της εφαρμογής, που φαίνεται στην Εικόνα 4.3. απεικονίζει πληροφορίες για τα τρία βασικά μετρούμενα μεγέθη (προπληρωμένα λεπτά ομιλία, προπληρωμένα γραπτά μηνύματα, προπληρωμένα δεδομένα διαδικτύου). Για κάθε ένα από τα μεγέθη αυτά υπάρχει τοξοειδής απεικόνιση, που εκφράζει την τρέχουσα τιμή ως προς την μέγιστη τιμή/τιμή αναφοράς. Στο κέντρο του κάθε τόξου απεικονίζεται το ποσοστό και αριθμητικά , κάτω δε από το τόξο υπάρχει το όνομα του μετρούμενου μεγέθους (Minutes, SMS, Data). Δεξιά του κάθε τόξου εμφανίζεται, σε μορφή κλάσματος, το τρέχον υπόλοιπο καθώς και η μέγιστη τιμή/τιμή αναφοράς. Στον ίδιο χώρο βρίσκεται το πλήκτρο που εμφανίζει ή κρύβει την ειδοποίηση (Toggle Notification) για το αντίστοιχο μέγεθος, καθώς και το πλήκτρο που επιτρέπει τον ορισμό της μέγιστης τιμής/τιμής αναφοράς.
Κάτω από αυτά τα τρία μπλόκ βρίσκεται το πεδίο Last Update που εμφανίζει την ημερομηνία και την ώρα του τελευταίου συγχρονισμού των απεικονιζόμενων μεγεθών, με τα αντίστοιχα του παρόχου. Κάτω από αυτό το πεδίο υπάρχει το πλήκτρο Update, που επιτρέπει στον χρήστη να στείλει γραπτό μήνυμα στον πάροχο ώστε να υπάρξει συγχρονισμός των μετρούμενων μεγεθών. Τέλος υπάρχει το πλήκτρο “HELP” που εμφανίζει την όψη Βοηθείας.



Εικόνα 4.3. Η βασική όψη της εφαρμογής.

4.4 Όψη ορισμού νέου μέγιστου/τιμής αναφοράς.

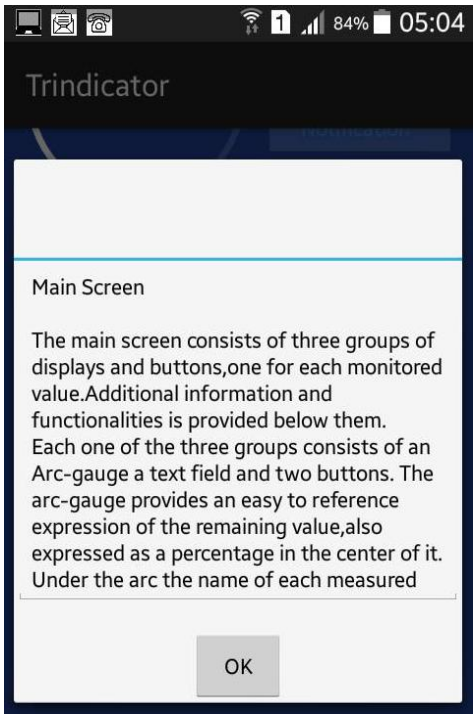
Πρόκειται όχι για μια αλλά για τρεις διαφορετικές όψεις (μια για κάθε παρεχόμενη υπηρεσία) που επιτρέπουν στον χρήστη να ορίσει την νέα τιμή αναφοράς/μέγιστη τιμή για το επιλεγμένο μέγεθος. Πέρα από το πεδίο ορισμού νέας τιμής ο χρήστης βλέπει την τρέχουσα τιμή καθώς και την τρέχουσα μέγιστη τιμή/τιμή αναφοράς.



Εικόνα 4.4. Όψη ορισμού μεγίστου.

4.5 Βοήθεια.

Στην όψη αυτή υπάρχουν οδηγίες για την λειτουργία της εφαρμογής. Ο χρήστης μπορεί να επιστρέψει στην προηγούμενη όψη με το πάτημα του OK ή του ανάλογου πλήκτρου επιστροφής του smartphone.



Εικόνα 4.5. Η όψη βοήθειας (απεικονίζει τμήμα των οδηγιών).

Κεφάλαιο 5
Ακρίβεια Δεδομένων.

Στο Κεφάλαιο αυτό αναλύεται το σημαντικό πρόβλημα της μη ταύτισης των τιμών των μετρουμένων μεγεθών (υπόλοιπο χρόνου ομιλίας, γραπτών μηνυμάτων, δεδομένων) μεταξύ του πελάτη και του παρόχου και παρουσιάζεται ο τρόπος αντιμετώπισής του.

5.1 Ακρίβεια Δεδομένων.

Ως ακρίβεια δεδομένων, για τις ανάγκες της παρούσας διπλωματικής εργασίας, ορίζεται η συνοχή και η ορθότητα των δεδομένων που έχουν στην διάθεσή τους εφαρμογή και χρήστης καθ' όλο τον κύκλο χρήσης και λειτουργίας της εφαρμογής. Σαν τιμή αναφοράς (reference) για τις τιμές των μετρούμενων μεγεθών ορίζεται η αντίστοιχη τιμή, όπως καταμετράται από τον πάροχο. Η ορθή λειτουργία της εφαρμογής προϋποθέτει την ταύτιση των τιμών των μετρουμένων μεγεθών, μεταξύ του πελάτη και του παρόχου.

Κάτι τέτοιο όμως είναι πρακτικά ανέφικτο σε ένα πραγματικό περιβάλλον, δεδομένης της πληθώρας τύπων των συσκευών, της υπάρχουσας τεχνολογίας χρονισμού μεταξύ παρόχου και συσκευής, των πολλαπλών πακέτων που μπορεί να προσφέρει ένας πάροχος, την περιορισμένη "γνώση" της συσκευής για την κατάσταση του δικτύου, τα πιθανά σφάλματα του χρήστη και την απρόβλεπτη συμπεριφορά της συσκευής (crash). Λόγω αυτών των συνθηκών κρίνεται αναγκαίο να επαναπροσδιοριστεί η έννοια της ακρίβειας ως η ελαχιστοποίηση τυχών αποκλίσεων μεταξύ χρήστη και παρόχου.

5.2 Ακρίβεια δεδομένων ως προς την φυσική σημασία των μεγεθών που εκφράζουν:

Οι τιμές που διατηρεί η εφαρμογή αντιστοιχούν σε μεγέθη με "φυσική" σημασία. Αυτό συνεπάγεται ότι το πεδίο τιμών αυτών των μεγεθών διέπεται από τους περιορισμούς των αντιστοίχων φυσικών μεγεθών. Είναι αναγκαίο, για την ορθή λειτουργία της εφαρμογής και για την επίτευξη ακρίβειας, οι περιορισμοί αυτοί να ληφθούν υπόψιν αλλά και να υπάρχουν τα μέσα για την διόρθωση πιθανής παραβίασης αυτών. Οι περιορισμοί και τα μέσα αναλύονται παρακάτω.

5.3 Ακρίβεια σε πολλαπλές ενεργοποιήσεις της εφαρμογής:

Βασική προϋπόθεση για την ομαλή λειτουργία της εφαρμογής είναι οι τιμές που διατηρεί και απεικονίζει να μην τροποποιούνται σε περίπτωση που ο χρήστης την καλέσει παραπάνω από

μια φορά. Με άλλα λόγια επιτρέπεται η ύπαρξη και εκτέλεση μόνο ενός ή κανενός στιγμιότυπου (instance) της εφαρμογής κάθε δεδομένη χρονική στιγμή. Εκκίνηση νέου επιτρέπεται μόνο όταν δεν υπάρχει ενεργό στιγμιότυπο την δεδομένη χρονική στιγμή. Διαφορετικά φέρεται στο προσκήνιο το τρέχον στιγμιότυπο.

5.4 Ακρίβεια δεδομένων μεταξύ τερματισμού και επόμενης ενεργοποίησης της εφαρμογής:

Οι τιμές των μετρούμενων μεγεθών αποθηκεύονται στην μνήμη του τηλεφώνου με τον τερματισμό της εφαρμογής και επαναφέρονται με την εκκίνησή της. Υπάρχει επίσης πρόβλεψη για την ορθότητα των μετρήσεων ακόμα και όταν η εφαρμογή είναι ανενεργή, όπως αναλύεται για κάθε μια των παρεχομένων υπηρεσιών στις παραγράφους 5.7 – 5.9.

5.5 Ακρίβεια δεδομένων σε απροσδόκητο τερματισμό της εφαρμογής (crash):

Για να εξασφαλιστεί η αποθήκευση των δεδομένων σε περίπτωση που υπάρχει κάποιος απρόσμενος τερματισμός της εφαρμογής έχει προβλεφθεί να γίνεται αυτόματη αποθήκευση των μετρούμενων μεγεθών στην μνήμη του τηλεφώνου κάθε δέκα λεπτά της ώρας.

5.6 Ακρίβεια γραφικού περιβάλλοντος - Graphic User Interface(GUI).

Η εφαρμογή δεν αρκεί να έχει στην διάθεσή της μεγέθη τα οποία είναι ορθά, πρέπει να έχει πρόσβαση σε αυτά και ο χρήστης. Ο χρήστης έχει πρόσβαση σε αυτά μέσω του γραφικού περιβάλλοντος, όπου απεικονίζονται κάποια από τα μεγέθη που διατηρεί η εφαρμογή. Για να πραγματοποιηθεί κάτι τέτοιο πρέπει να ανανεώνεται το γραφικό περιβάλλον σύμφωνα με τις νέες τιμές. Η ανανέωση αυτή γίνεται τόσο ανά τακτά χρονικά διαστήματα όσο και όταν γίνεται παρατήρηση συγκεκριμένων γεγονότων. Συγκεκριμένα, το γραφικό περιβάλλον της εφαρμογής ανανεώνεται κάθε λεπτό της ώρας αλλά και όποτε συμβαίνει κάτι από τα παρακάτω:

- Εκκίνηση της εφαρμογής.
- Εμφάνιση της εφαρμογής στο “προσκήνιο” (foreground).
- Τερματισμός τηλεφωνικής κλήσης.
- Ορισμός νέας τιμής αναφοράς.

5.7 Ακρίβεια προπληρωμένου χρόνου ομιλίας.

Η εφαρμογή μετρά σε λεπτά της ώρας τον υπολειπόμενο προπληρωμένο χρόνο ομιλίας. Επίσης υπάρχει και ένα μέγεθος αναφοράς, που ο χρήστης έχει την δυνατότητα να ορίσει, το οποίο εκφράζει τον αρχικό/μέγιστο χρόνο ομιλίας που διαθέτει. Ο υπολειπόμενος χρόνος εκφράζεται και ως ποσοστό επί τοις εκατό με βάση τα δύο προαναφερθέντα μεγέθη. Με βάση την ποσοστιαία τιμή ορίζεται και η τιμή του αντίστοιχου τοξοειδούς ενδείκτη. Η εφαρμογή υπολογίζει τα πόσα λεπτά υπολείπονται μετρώντας την διάρκεια κάθε εξερχόμενης κλήσης και με τον τερματισμό αυτής αφαιρεί την διάρκειά της από τα υπολειπόμενα λεπτά. Ως διάρκεια κλήσης ορίζουμε τον χρόνο από την έναρξη της συνομιλίας ως τον τερματισμό αυτής, στρογγυλοποιημένο προς τον επόμενο ακέραιο. Σε περίπτωση που η εφαρμογή δεν έχει εκκινήσει, με την έναρξη κάθε κλήσης ενεργοποιείται αυτομάτως προκειμένου να πραγματοποιήσει την μέτρηση.

Ισχύουν οι ακόλουθοι περιορισμοί ως προς την φυσική σημασία του προπληρωμένου χρόνου ομιλίας.

- Ο υπολειπόμενος προπληρωμένος χρόνος δεν μπορεί να είναι αρνητικός. Αν πάρει αρνητική τιμή τότε δεν υπολείπεται προπληρωμένος χρόνος και η μεταβλητή γίνεται μηδέν.
- Η μέγιστη τιμή/τιμή αναφοράς προπληρωμένου χρόνου ομιλίας δεν μπορεί να είναι αρνητική. Σε περίπτωση που παρθεί τέτοια τιμή αυτόματα ορίζεται ο τρέχων υπολειπόμενος προπληρωμένος χρόνος ως χρόνος αναφοράς.
- Η μέγιστη τιμή/τιμή αναφοράς δεν μπορεί να είναι μικρότερη από τον υπολειπόμενο προπληρωμένο χρόνο. Σε περίπτωση που παρθεί τέτοια τιμή, αυτόματα ορίζεται ο τρέχων υπολειπόμενος προπληρωμένος χρόνος ως χρόνος αναφοράς.
- Συνέπεια των παραπάνω είναι ότι η ποσοστιαία έκφραση του υπολειπόμενου προπληρωμένου χρόνου ομιλίας κυμαίνεται μεταξύ 0%-100%.

5.8 Ακρίβεια προπληρωμένων γραπτών μηνυμάτων.

Η εφαρμογή παρακολουθεί την κίνηση γραπτών μηνυμάτων. Υπάρχει επίσης και μέγεθος αναφοράς το οποίο εκφράζει το αρχικό/μέγιστο πλήθος προπληρωμένων γραπτών μηνυμάτων το οποίο ο χρήστης μπορεί να ορίσει. Τέλος υπάρχει και εδώ απεικόνιση ως ποσοστό επί τοις εκατό με βάση τα δυο προηγούμενα μεγέθη. Με βάση την ποσοστιαία τιμή ορίζεται και η τιμή του αντίστοιχου τοξοειδούς ενδείκτη.

Ο τρόπος που η εφαρμογή λειτουργεί είναι ότι κάθε φορά που στέλνεται επιτυχώς κάποιο γραπτό μήνυμα ο μετρητής μειώνεται κατά ένα, εφόσον η εφαρμογή είναι σε λειτουργία.

Όποτε πραγματοποιείται εκκίνηση της εφαρμογής από τον χρήστη, η εφαρμογή εξετάζει τις εγγραφές των εξερχόμενων μηνυμάτων. Καταμετρά πόσες έγιναν από την τελευταία φορά που ήταν σε λειτουργία και αφαιρεί τον αριθμό αυτόν από το πλήθος των προπληρωμένων γραπτών μηνυμάτων.

Ισχύουν οι ακόλουθοι περιορισμοί ως προς την φυσική σημασία των προπληρωμένων γραπτών μηνυμάτων.

- Τα υπολειπόμενα προπληρωμένα γραπτά μηνύματα δεν μπορεί να είναι αρνητικά. Αν ο μετρητής πάρει αρνητική τιμή τότε δεν υπολείπονται γραπτά μηνύματα και η μεταβλητή ορίζεται ως μηδενική.
- Η μέγιστη τιμή/τιμή αναφοράς προπληρωμένων γραπτών μηνυμάτων δεν μπορεί να είναι αρνητική. Σε περίπτωση που παρθεί τέτοια τιμή αυτόματα ορίζεται ως τρέχουσα τιμή αναφοράς το τρέχον υπόλοιπο γραπτών μηνυμάτων.
- Η μέγιστη τιμή/τιμή αναφοράς δεν μπορεί να είναι μικρότερη από τον υπολειπόμενο προπληρωμένο χρόνο. Σε περίπτωση που παρθεί τέτοια τιμή αυτόματα ορίζεται ως τρέχουσα τιμή αναφοράς το τρέχον υπόλοιπο γραπτών μηνυμάτων.
- Συνέπεια των παραπάνω είναι ότι η ποσοστιαία έκφραση των υπολειπόμενων προπληρωμένων μηνυμάτων κυμαίνεται μεταξύ 0%-100%.

5.9 Ακρίβεια προπληρωμένων δεδομένων διαδικτύου.

Η εφαρμογή παρακολουθεί την χρήση προπληρωμένων megabyte που κάνει η συσκευή. Στις μετρήσεις αυτές αγνοεί τα δεδομένα τα οποία δεν χρεώνουν τον χρήστη (δίκτυα wi-fi). Η εφαρμογή κάνει χρήση της υπάρχουσας δυνατότητας που προσφέρουν οι συσκευές με λειτουργικό σύστημα android. Κάθε ένα λεπτό της ώρας γίνεται έλεγχος και η εφαρμογή ανανεώνει το υπόλοιπο αφαιρώντας την ανάλογη χρήση δεδομένων διαδικτύου που έχει γίνει. Υπάρχει και εδώ, όπως και στα προηγούμενα μεγέθη, μέγιστη τιμή/τιμή αναφοράς, την οποία μπορεί να αλλάξει ο χρήστης. Τέλος και εδώ εκφράζεται το υπόλοιπο και σαν ποσοστό επί τοις εκατό. Με βάση την ποσοστιαία τιμή ορίζεται και η τιμή του αντίστοιχου τοξοειδούς ενδείκτη.

Ισχύουν οι ακόλουθοι περιορισμοί ως προς την φυσική σημασία των προπληρωμένων δεδομένων διαδικτύου.

- Τα υπολειπόμενα προπληρωμένα δεδομένα δεν μπορεί να είναι αρνητικά. Αν ο μετρητής πάρει αρνητική τιμή, τότε δεν υπολείπονται προπληρωμένα megabyte και η μεταβλητή (το υπόλοιπο) ορίζεται ως μηδενική.

- Η μέγιστη τιμή/τιμή αναφοράς προπληρωμένων δεδομένων διαδικτύου δεν μπορεί να είναι αρνητική. Σε περίπτωση που παρθεί τέτοια τιμή αυτόματα ορίζεται ως τρέχουσα τιμή αναφοράς το τρέχον υπόλοιπο δεδομένων διαδικτύου.
- Η μέγιστη τιμή/τιμή αναφοράς δεν μπορεί να είναι μικρότερη από τα υπολειπόμενα πακέτα διαδικτύου. Σε περίπτωση που παρθεί τέτοια τιμή αυτόματα ορίζεται ως τρέχουσα τιμή αναφοράς το τρέχον υπόλοιπο δεδομένων διαδικτύου.
- Συνέπεια των παραπάνω είναι ότι η ποσοστιαία έκφραση των υπολειπόμενων δεδομένων διαδικτύου κυμαίνεται μεταξύ 0%-100%.

5.10 Ακρίβεια ειδοποιήσεων (notifications).

Η εφαρμογή παρέχει την δυνατότητα στον χρήστη να εμφανίζει στο πεδίο ειδοποιήσεων της συσκευής εικονίδια που αντιπροσωπεύουν το υπόλοιπο για μια ή περισσότερες από τις μετρούμενες ποσότητες (Προπληρωμένος χρόνος ομιλίας, προπληρωμένα γραπτά μηνύματα, και προπληρωμένα δεδομένα διαδικτύου). Επίσης δίνεται η δυνατότητα στον χρήστη να δει περισσότερες πληροφορίες για τα μεγέθη αυτά, επεκτείνοντας το πεδίο ειδοποιήσεων καθώς και να μεταβεί στην εφαρμογή. Η ακρίβεια των τιμών που απεικονίζονται στις ειδοποιήσεις είναι συνέπεια της ακρίβειας του γραφικού περιβάλλοντος, της ακρίβειας του προπληρωμένου χρόνου, της ακρίβειας των προπληρωμένων γραπτών μηνυμάτων καθώς και της ακρίβειας δεδομένων διαδικτύου.

5.11 Επαλήθευση και διόρθωση των μετρούμενων μεγεθών.

Λόγω των περιορισμών της εφαρμογής αλλά και των ιδιαιτεροτήτων των τηλεφωνικών δικτύων και των συνδέσεων διαδικτύου είναι αδύνατο οι τιμές που διατηρεί η εφαρμογή να είναι ακριβώς ίδιες με τις ανάλογες που διατηρεί ο πάροχος. Αν και τα σφάλματα αυτά είναι αρχικά αμελητέα, εύκολα μετατρέπονται σε σοβαρές αποκλίσεις, οι οποίες μπορούν να προκαλέσουν σύγχυση στον χρήστη και να παραβιάσουν κάθε έννοια ακρίβειας δεδομένων. Για αυτό τον λόγο η εφαρμογή αποσκοπεί στην διόρθωση αυτών των αποκλίσεων διασταυρώνοντας τις τιμές με τον πάροχο και συγχρονίζοντας αυτές με τις τιμές του παρόχου. Αυτό γίνεται αυτόματα κάθε δύο ωρολογιακές ώρες και επίσης δίνεται στον χρήστη η δυνατότητα να κάνει ο ίδιος συγχρονισμό αυτοβούλως. Η μέθοδος που χρησιμοποιείται είναι ή αποστολή γραπτού μηνύματος στον ειδικό τηλεφωνικό αριθμό του παρόχου, ο οποίος με την σειρά του με γραπτό μήνυμα ενημερώνει τον χρήστη για το υπόλοιπο του σε προπληρωμένα λεπτά ομιλίας, προπληρωμένα γραπτά μηνύματα και πακέτα διαδικτύου. Η εφαρμογή διαβάζει το μήνυμα και ανανεώνει τις ανάλογες τιμές.

Κεφάλαιο 6

Σχολιασμός Δομικών Στοιχείων της Εφαρμογής.

Στο κεφάλαιο αυτό παρουσιάζονται αποσπάσματα από τον κώδικα της εφαρμογής που κρίνεται αναγκαίο να σχολιαστούν, είτε για λόγους επεξήγησης από την προγραμματιστική σκοπιά είτε γιατί αποτελούν απόδειξη της ορθότητάς της , ως υλοποίηση των κριτηρίων ακρίβειας, που προσδιορίστηκαν στο προηγούμενο κεφάλαιο. Διευκρινίζεται ότι ο πλήρης κώδικας της εφαρμογής βρίσκεται στο Παράρτημα Α.

6.1. Ο Κώδικας java.

Στις ακόλουθες παραγράφους παρουσιάζονται αποσπάσματα απο τον κώδικα java της εφαρμογής. Αξίζει να σημειωθεί ότι η ανάπτυξη μιας εφαρμογής android διέπεται από αρκετές ιδιαιτερότητες που δεν απαντώνται γενικά στην java.

6.1.1. Εκκίνηση-Συνέχεια-Τερματισμός της εφαρμογής.

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      bootTime=System.currentTimeMillis()- SystemClock.elapsedRealtime();
      Log.e("BOOT TIME:",Long.toString(bootTime));
      activity=this;
      setContentView(R.layout.activity_main);
      context=getApplicationContext();
      SharedPreferences sharedPreferences= getPreferences(MODE_PRIVATE);
      minutes=sharedPreferences.getInt("minutes",0);
      minutesMax=sharedPreferences.getInt("minutesMax",100);
      sms=sharedPreferences.getInt("sms",0);
      smsMax=sharedPreferences.getInt("smsMax",100);
      data=sharedPreferences.getInt("data",0);
      dataMax=sharedPreferences.getInt("dataMax",100);
      hasUpdated=sharedPreferences.getBoolean("hasUpdated", false);
      lastMessage=sharedPreferences.getString("MainActivitylastMessage", "-a");
      thisMessage=sharedPreferences.getString("MainActivitythisMessage",null);
```

```
        lastDataCheck=sharedPreferences.getLong("lastDataCheck",0);
        lastUpdateMS=sharedPreferences.getLong("lastUpdateMS",0);

previousTotalBytes=sharedPreferences.getLong("previousTotalBytes",MyTrafficStats.getM
obileTxBytes());
        if (bootTime>lastDataCheck)
        {
           previousTotalBytes=0;
        }
        if (hasUpdated==true) {
           UpdateDate=sharedPreferences.getString("UpdateDate","");
        }
        Handler handle = new Handler(){};
        myObserver = new SentSMSObserver(handle);
            if (hasUpdated==true) {
             myObserver.CheckPast();
        }
        contentResolver = getContentResolver();
        contentResolver.registerContentObserver(Uri.parse("content://sms"),true,
myObserver);
        Log.d("onCreate", "OBSERVER ON");
        activityReference=this;
    }
```

Με την εκκίνηση της εφαρμογής το σύστημα υπολογίζει τον χρόνο που έγινε η εκκίνηση της συσκευής. Έπειτα εμφανίζει την κεντρική όψη της εφαρμογής. Για να γίνει αυτό ελέγχει για το αν υπάρχουν αποθηκευμένες τιμές από προηγούμενη λειτουργία της εφαρμογής. Αν υπάρχουν γίνεται ανάκτηση των τιμών αυτών. Αν δεν υπάρχουν δίνονται κάποιες default τιμές στις μεταβλητές. Τα μεγέθη τα οποία ανακαλούνται και ανακτώνται είναι οι τρέχουσες τιμές και οι τρέχουσες τιμές/τιμές αναφοράς για τα τρία μετρούμενα μεγέθη (προπληρωμένα λεπτά,μηνύματα,δεδομένα διαδικτύου). Επίσης τηρούνται οι τιμές για μεγέθη τα οποία χρειάζονται για την παρακολούθηση των δεδομένων διαδικτύου αλλά και για τα γραπτά μηνύματα. Η χρήση αυτών εξηγείται στην ανάλογη παράγραφο. Ορίζεται επίσης ο content observer με την βοήθεια του οποίο παρακολουθούνται τα γραπτά μηνύματα. Αξίζει να

σημειωθεί η πρώτη εντολή που εκτελείται, super.onCreate(savedInstanceState) είναι κλήση στην αντίστοιχη μέθοδο onCreate της υπέρ-κλάσης activity. Με αυτό τον τρόπο εξασφαλίζεται ομοιογένεια στον κύκλο λειτουργίας των εφαρμογών android.

```
@Override
   public void onResume() {
      super.onResume();
   handlerGUI.removeCallbacks(runnableCode1);
   handlerData.removeCallbacks(runnableCode2);
   UpdateGUI();
   activityReference=this;
   handlerStoreAndUpdate.post(runnableCode1);
   handlerDataAndGUI.post(runnableCode2);
   }
```

Με την συνέχεια της εφαρμογής σταματάν τα threads που εκτελούνται και ξεκινάνε πάλι. Αυτό γίνεται προκειμένου να υπάρχει μόνο ένα στιγμιότυπο του κάθε νήματος. Επίσης, γίνεται ανανέωση του γραφικού περιβάλλοντος της εφαρμογής. Όπως και πριν έτσι και εδώ γίνεται κλήση στην μέθοδο onResume() της υπερ-κλάσσης.

```
@Override
   protected void onDestroy(){
      super.onDestroy();
      Store();
      minutesToggle=false;
      MinutesNotification(0);
      smsToggle=false;
      SMSNotification(0);
      dataToggle=false;
      DataNotification(0);
      getContentResolver().unregisterContentObserver(myObserver);
   }
```

Εδώ ορίζεται η συμπεριφορά της εφαρμογής όταν τερματίζεται ομαλώς. Όμοια με πριν

γίνεται κλήση της μεθόδου onDestroy() της υπερ-κλάσσης. Μετά με την μέθοδο Store() που αναλύεται παρακάτω αποθηκεύονται οι μεταβλητές που ανακαλούνται κατά την εκκίνησης της εφαρμογής. Στην συνέχεια απενεργοποιούνται οι ειδοποιήσεις και απενεργοποιείται ο contentObserver που χρησιμοποιείται για την παρακολούθηση των προπληρωμένων γραπτών μηνυμάτων.

6.1.2. Αποθήκευση-Ανάκτηση Δεδομένων.

Αναφέρθηκε σε προηγούμενη παράγραφο η ανάγκη για ακρίβεια των δεδομένων μεταξύ τερματισμού της εφαρμογής και μεταγενέστερης ενεργοποίησης. Για αυτό τον λόγο χρησιμοποιούμε την κλάση SharedPreferences. Η κλάση αυτή μας επιτρέπει να αποθηκεύουμε ζεύγη key-value,δηλαδή ονόματος και τιμής. Τα ζεύγη αυτά αποθηκεύονται στην μνήμη της συσκευής και δεν χάνονται, εκτός αν ο χρήστης επιλέξει να τα διαγράψει χειροκίνητα. Στην εφαρμογή η χρήση τους γίνεται στο onCreate() που η εφαρμογή ανακτά τα δεδομένα και στην μέθοδο Store() που η εφαρμογή αποθηκεύει τα δεδομένα αυτά. Αξίζει να σημειωθεί ότι η μέθοδος Store() δεν καλείται μόνο στον τερματισμό της εφαρμογής αλλά ανά τακτά χρονικά διαστήματα προκειμένου να εξασφαλιστεί προστασία των δεδομένων, ακόμα και σε περίπτωση μη ομαλού τερματισμού της εφαρμογής (crash), όπως ορίστηκε το ανάλογο κριτήριο ακρίβειας (Παράγραφος 5.5 Ακρίβεια δεδομένων σε απροσδόκητο τερματισμό της εφαρμογής). Τέλος σημειώνεται ότι σε περίπτωση που η εφαρμογή ξεκινήσει πρώτη φορά σε συσκευή που δεν υπάρχουν δεδομένα προηγούμενης αποθήκευσης ή αν ο χρήστης τα έχει διαγράψει χειροκίνητα οι μεταβλητές παίρνουν κάποιες default τιμές.

Παράδειγμα: Η μεταβλητή minutes κρατάει το υπόλοιπο των προπληρωμένων λεπτών ομιλίας. Με την εκκίνηση της εφαρμογής εκτελείται η ακόλουθη εντολή για ανάκτηση:
minutes=sharedPreferences.getInt("minutes",0);
Με τον τερματισμό εκτελείται η παρακάτω εντολή:
editor.putInt("minutes",minutes);

Εδώ στην ουσία ορίζουμε ένα key-value ζεύγος. Το key για λόγους ευκολίας έχει το ίδιο όνομα με την μεταβλητή που αποθηκεύει (“minutes”). Με την εντολή :
minutes= sharedPreferences.getInt("minutes",0); γίνεται ανάκτηση της τιμής, ενώ αν δεν υπάρχει ανάλογο προαποθηκευμένο key-value pair έχουμε ορίσει default τιμή 0.

6.1.3. Παρακολούθηση Λεπτών.

Η παρακολούθηση της χρήσης των λεπτών γίνεται σε πραγματικό χρόνο κατά την διάρκεια της κλήσης. Η βασική ιδέα είναι ότι η εφαρμογή κρατά τον χρόνο εκκίνησης της κλήσης και τον χρόνο τερματισμού όταν η κλήση ολοκληρωθεί. Η διαφορά αυτών είναι προφανώς η χρονική διάρκεια της κλήσης. Η χρέωση γίνεται ανά λεπτό με στρογγυλοποίηση προς τα πάνω και ελάχιστη χρέωση ένα λεπτό. Για την πραγματοποίηση αυτής της λειτουργίας χρησιμοποιούμε δύο δικές μας κλάσεις. Η CustomPhoneStateListener και η PhoneStateBroadcastReceiver.

6.1.3.1 Η κλάση CustomPhoneStateListener.

```
public class CustomPhoneStateListener extends PhoneStateListener {
    private final String TAG = "PhoneStateListener";

    @Override
    public void onCallStateChanged(int state, String incomingNumber){
      if( incomingNumber != null && incomingNumber.length() > 0 )
        incoming_number = incomingNumber;

      switch(state){
        case TelephonyManager.CALL_STATE_RINGING:
          prev_state=state;
          break;

        case TelephonyManager.CALL_STATE_OFFHOOK:
          if ((isOn==0)&&(prev_state!=TelephonyManager.CALL_STATE_RINGING))
{
            isOn=1;
            prev_state=state;
            callStart= Calendar.getInstance();
          }
          break;

        case TelephonyManager.CALL_STATE_IDLE:
          if((prev_state == TelephonyManager.CALL_STATE_OFFHOOK)){
```

```
                if (isOn==1){
                    isOn=0;
                    prev_state=state;
                    callEnd= Calendar.getInstance();
                    callDuration= (callEnd.getTimeInMillis()-callStart.getTimeInMillis())/
(1000);
                    callDuration=1+callDuration/60;
                    if (callDuration>=MainActivity.minutes){
                       MainActivity.minutes=0;
                    }
                    MainActivity.minutes= (int) (MainActivity.minutes-callDuration);
                    MainActivity.minutesPercent=
(MainActivity.minutes*100/MainActivity.minutesMax);
                    Intent startIntent = new Intent(mContext, MainActivity.class);
                    startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    mContext.startActivity(startIntent);
                }
                }
                if((prev_state == TelephonyManager.CALL_STATE_RINGING)){
                    prev_state=state;
                }
                break;
            }
        }
    }
```

Για την κατανόηση της λειτουργίας της κλάσης αυτής απαιτείται μια σύντομη αναφορά στις καταστάσεις του τηλεφώνου (Call States).

Υπάρχουν τρεις καταστάσεις στις οποίες μπορεί να βρίσκεται το τηλέφωνο:

- Idle: Δεν υπάρχει κλήση.
- Ringing: Εισερχόμενη κλήση, είτε το τηλέφωνο χτυπάει είτε η κλήση μπαίνει σε αναμονή επειδή υπάρχει ήδη άλλη ενεργή κλήση.
- Off-hook: Υπάρχει κλήση, είτε ενεργή, είτε ξεκινάει τώρα, είτε σε αναμονή και δεν υπάρχει εισερχόμενη κλήση.

Η κλάση CustomPhoneStateListener παρατηρεί σε ποία κατάσταση βρίσκεται το τηλέφωνο και λειτουργεί με βάση την τρέχουσα και την προηγούμενη κατάσταση. Καλείται κάθε φορά που υπάρχει αλλαγή κατάστασης.

Εκκίνηση εξερχόμενης κλήσης υπάρχει όταν η κατάσταση του τηλεφώνου είναι Off-hook και η προηγούμενη κατάσταση δεν είναι Ringing. Αντίστοιχα, τερματισμός κλήσης υπάρχει όταν η κατάσταση του τηλεφώνου είναι idle και η προηγούμενη είναι Off-hook. Επειδή η αλλαγή κατάστασης μπορεί να καλέσει παραπάνω από μια φορά την μέθοδο onCallStateChanged χρησιμοποιούμε την μεταβλητή isOn σαν δυαδικό σημαφόρο (binary semaphore).

Κατά τον τερματισμό της κλήσης γίνεται έλεγχος ακρίβειας δεδομένων των υπολειπόμενων προπληρωμένων λεπτών ομιλίας και η εφαρμογή έρχεται στο προσκήνιο.

6.1.3.2 Η κλάση PhoneStateBroadcastReceiver.

```
package kozos.trindicator;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import java.util.Calendar;

public class PhoneStateBroadcastReceiver extends BroadcastReceiver {

   private final String TAG = "PhoneState";
   Communicator comm2;
   static Context mContext;
   String incoming_number;
   private static int prev_state;
   static int isOn=0;
   public String phoneNumber="1";
   public String ID;
```

```
    public static Calendar callEnd,callStart;
    long callDuration;

    @Override
    public void onReceive(Context context, Intent intent) {
        mContext = context;
        Intent startIntent = new Intent(mContext, MainActivity.class);
        startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(startIntent);
        comm2=(Communicator)MainActivity.activity;
        TelephonyManager telephony =
(TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
        CustomPhoneStateListener customPhoneListener = new CustomPhoneStateListener();
        telephony.listen(customPhoneListener, PhoneStateListener.LISTEN_CALL_STATE);
        ID= telephony.getSimOperator();
        Bundle bundle = intent.getExtras();
        phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
        String phoneNr = bundle.getString("incoming_number");
    }
```

Η κλάση PhoneStateBroadcastReceiver πρακτικά αποτελεί το μέσο με το οποίο δημιουργείται αντικείμενο της κλάσης CustomPhoneStateListener. Κάθε φορά που υπάρχει broadcast σχετιζόμενο με κλήσεις η εφαρμογή έρχεται στο προσκήνιο.

6.1.4. Παρακολούθηση Γραπτών Μηνυμάτων.

Η παρακολούθηση της χρήσης των προπληρωμένων γραπτών μηνυμάτων γίνεται με την βοήθεια της κλάσης SentSMSObserver. Η κλάση αυτή αποτελεί επέκταση της κλάσης ContentObserver. Η τηλεφωνική συσκευή αποθηκεύει πληροφορίες για τα γραπτά μηνύματα που στέλνει και λαμβάνει σε ειδικούς πίνακες (content providers). Η κλάση ασχολείται με τον πίνακα που αποθηκεύει τα σταλθέντα μηνύματα και πληροφορίες για αυτά (content://sms/sent).

Η λειτουργία της κλάσης βρίσκεται στις δύο μεθόδους: onChange() και CheckPast().

6.1.4.1. Η μέθοδος onChange()

```
@Override
   public void onChange(boolean selfChange){
      super.onChange(selfChange);
      Uri smsuri = Uri.parse("content://sms/sent");
      Cursor cursor = MainActivity.context.getContentResolver().query(smsuri, null, null,
null, null);
      String outgoingSMS = null;
      if (cursor != null && cursor.moveToFirst()){
         try{
            cursor.moveToFirst();
            String type = cursor.getString(cursor.getColumnIndex("type"));
            MainActivity.thisMessage = cursor.getString(cursor.getColumnIndex("_id"));
            if (type.equals("2")&&(!
(MainActivity.thisMessage.equals(MainActivity.lastMessage)))){
               if (MainActivity.sms>1) {
                  MainActivity.sms=MainActivity.sms-1;
                  if ((MainActivity.sms<=10)&&(MainActivity.sms>1)){
                     Toast.makeText(MainActivity.context, "Running Out Of free SMS",
Toast.LENGTH_LONG).show();
                  }
                  if (MainActivity.sms==0){
                     Toast.makeText(MainActivity.context, "You just run out of Free
SMS",Toast.LENGTH_LONG).show();
                  }
               }else if(MainActivity.sms==0){
                  Toast.makeText(MainActivity.context, "You have run out of Free
SMS",Toast.LENGTH_LONG).show();
               }

               MainActivity.sms=Math.max(MainActivity.sms,0);
               MainActivity.smsPercent=(100*MainActivity.sms)/MainActivity.smsMax;
               MainActivity.lastMessage = MainActivity.thisMessage;
```

```
            }
        }
        catch (CursorIndexOutOfBoundsException e){
        }
        finally{
            cursor.close();
        }

        if (outgoingSMS != null ){
          Log.d("OBSERVER", "SMSHelper: outgoingSMS: " + outgoingSMS);
        }
    }
}
```

Ο content provider προσφέρει αρκετές πληροφορίες για κάθε απεσταλμένο γραπτό μήνυμα. Η εφαρμογή απασχολείται μόνο με τα γραπτά μηνύματα τα οποία έχουν τύπο 2 (type 2), αυτά δηλαδή που έχουν σταλεί επιτυχώς. Επίσης κάθε sms έχει ένα διαφορετικό χαρακτηριστικό/ταυτότητα (id). Η μέθοδος onChange καλείται κάθε φορά που υπάρχει κάποια αλλαγή στον πίνακα. Αρκεί λοιπόν να δούμε ότι η νέα εγγραφή στον πίνακα σταλμένων μηνυμάτων έχει id διαφορετικό από τον προηγούμενο και είναι τύπου 2.

6.1.4.2. Η μέθοδος CheckPast()

```
   public void CheckPast(){
      int i=0;
      String pastType,pastId;

      Uri smsuri = Uri.parse("content://sms/sent");
      Cursor cursor = MainActivity.context.getContentResolver().query(smsuri, null, null,
null, null);
      if (!(MainActivity.lastMessage.equals("-
a")&&(MainActivity.thisMessage.equals(null)))) {
            cursor.moveToFirst();
            while(!
(cursor.getString(cursor.getColumnIndex("_id")).equals(MainActivity.lastMessage))) {
```

```
                if (cursor.getString(cursor.getColumnIndex("type")).equals("2")){
                i++;
                }
            cursor.moveToNext();
            }
        cursor.moveToFirst();
        MainActivity.lastMessage = cursor.getString(cursor.getColumnIndex("_id"));
        MainActivity.sms=MainActivity.sms-i;
        MainActivity.sms=Math.max(MainActivity.sms,0);
        MainActivity.smsPercent=(100*MainActivity.sms)/MainActivity.smsMax;
        if ((MainActivity.sms<=10)&&(MainActivity.sms>1)){
            Toast.makeText(MainActivity.context, "Running Out Of free SMS",
Toast.LENGTH_LONG).show();
        }
        if(MainActivity.sms==0){
            Toast.makeText(MainActivity.context, "You have run out of Free
SMS",Toast.LENGTH_LONG).show();
        }
        }
    }
```

Η μέθοδος CheckPast() καλείται κατά την εκκίνηση της εφαρμογής. Απλά μετρά όλα τα απεσταλμένα μηνύματα τύπου 2 από την τελευταία φορά που η εφαρμογή ήταν σε λειτουργία και αφαιρεί το πλήθος από το υπόλοιπο των προπληρωμένων γραπτών μηνυμάτων.

Και στις δύο περιπτώσεις γίνεται έλεγχος προκειμένου να ισχύουν οι περιορισμοί που αναφέρθηκαν στις παραγράφους 5.3 (Ακρίβεια σε πολλαπλές ενεργοποιήσεις της εφαρμογής) και 5.8 (Ακρίβεια προπληρωμένων γραπτών μηνυμάτων).

6.1.5. Παρακολούθηση Πακέτων Διαδικτύου.

```
    Handler handlerDataAndGUI = new Handler();


    private Runnable runnableCode2 = new Runnable() {
        @Override
```

```
    public void run() {
    res=MyTrafficStats.getMobileTxBytes();
       lastDataCheck=System.currentTimeMillis();
    if (res-previousTotalBytes>1000000L-1L){
       reducer=(int)((res-previousTotalBytes)/1000000L);
       data=data-reducer;
       data=Math.max(data,0);
       previousTotalBytes=previousTotalBytes+(long)reducer*1000000L;
       UpdateGUI();
    }
       handlerDataAndGUI.postDelayed(runnableCode2, 1*60000);
    }
};
```

Τα πακέτα διαδικτύου παρακολουθούνται με τον παραπάνω κώδικα. Πρόκειται περί thread το οποίο εκτελείται κάθε λεπτό. Η υλοποίηση βασίζεται σε μεγάλο βαθμό στην μέθοδο getMobileTxBytes που μετρά το σύνολο των bytes που έχουν σταλεί ή έχουν ληφθεί από την συσκευή από την ώρα εκκίνησης. Σε περίπτωση που έχει γίνει νέα εκκίνηση της εφαρμογής λαμβάνεται υπόψιν στις μετρήσεις που γίνονται στην μέθοδο onCreate() παραπάνω. Όπως και πριν υπάρχει μέριμνα προκειμένου τα μεγέθη να μην πάρουν τιμές που δεν αντιστοιχούν σε πραγματικές.

6.1.6. Ανανέωση Γραφικού Περιβάλλοντος.

```
public void UpdateGUI(){
    ArcProgress ap = (ArcProgress) findViewById(R.id.arcMinutes);
    TextView tv=(TextView) findViewById(R.id.textMinutesDisplay);
    minutes=Math.max(minutes,0);
    minutesMax=Math.max(Math.max(minutes,minutesMax),100);
    minutesPercent=(100*minutes)/minutesMax;
    ap.setProgress(minutesPercent);
    tv.setText(" " + Integer.toString(minutes) + " / " + Integer.toString(minutesMax) + " ");
    ap = (ArcProgress) findViewById(R.id.arcSMS);
    tv=(TextView) findViewById(R.id.textSMSDisplay);
    sms=Math.max(sms,0);
```

```
        smsMax=Math.max(Math.max(sms,smsMax),100);
        smsPercent=(100*sms)/smsMax;
        ap.setProgress(smsPercent);
        tv.setText(" " + Integer.toString(sms) + " / " + Integer.toString(smsMax) + " ");
        ap = (ArcProgress) findViewById(R.id.arcData);
        tv=(TextView) findViewById(R.id.textDataDisplay);
        data=Math.max(data,0);
        dataMax=Math.max(Math.max(data,dataMax),100);
        dataPercent=(100*data)/dataMax;
        ap.setProgress(dataPercent);
        tv.setText(" "+Integer.toString(data)+" / "+Integer.toString(dataMax)+" ");
        if (hasUpdated==true){
            tv=(TextView) findViewById(R.id.textLastUpdate);
            tv.setText(" Last Update: "+UpdateDate+" ");
        }else{
            tv=(TextView) findViewById(R.id.textLastUpdate);
            tv.setText("  PENDING  ");
        }
        if (minutesToggle==true){
            MinutesNotification(1);
        }
        if (smsToggle==true){
            SMSNotification(1);
        }
        if (dataToggle==true){
            DataNotification(1);
        }
    }
```

Η μέθοδος UpdateGUI() είναι ίσως η πλέον σημαντική μέθοδος της εφαρμογής. Καλείται περιοδικά ανά λεπτό της ώρας ή μετά από συγκεκριμένα γεγονότα. Εκτός από την ακρίβεια γραφικού περιβάλλοντος εξασφαλίζει και την ακρίβεια των δεδομένων προπληρωμένου χρόνου, προπληρωμένων γραπτών μηνυμάτων, και προπληρωμένων δεδομένων διαδικτύου. Συνοπτικά για τα τρία βασικά μεγέθη η λογική είναι ίδια. Θα δειχθεί τι γίνεται για τα

προπληρωμένα λεπτά ομιλίας και η διαδικασία είναι όμοια για τα υπόλοιπα μεγέθη.

```
1.      ArcProgress ap = (ArcProgress) findViewById(R.id.arcMinutes);
2.      TextView tv=(TextView) findViewById(R.id.textMinutesDisplay);
3.      minutes=Math.max(minutes,0);
4.      minutesMax=Math.max(minutes,minutesMax)
5.      minutesPercent=(100*minutes)/minutesMax;
6.      ap.setProgress(minutesPercent);
7.      tv.setText(" " + Integer.toString(minutes) + " / " + Integer.toString(minutesMax) + "
");

{...}

i.          if (minutesToggle==true){
ii.         MinutesNotification(1);
```

Στην πρώτη γραμμή η εφαρμογή βρίσκει την τοξοειδή απεικόνιση που αντιστοιχεί στα λεπτά και στην δεύτερη το πεδίο κειμένου που αναφέρεται σε αυτά. Στην τρίτη και την τέταρτη γραμμή εξασφαλίζεται η ακρίβεια των δεδομένων ως προς την φυσική τους σημασία, όπως έχει αναφερθεί στη παράγραφο 5.2 (Ακρίβεια δεδομένων ως προς την φυσική σημασία των μεγεθών που εκφράζουν), δηλαδή τα υπολειπόμενα λεπτά πρέπει να είναι θετικά ή μηδέν και μικρότερα ή ίσα της μέγιστης τιμής/τιμής αναφοράς. Στην πέμπτη γραμμή υπολογίζεται η ποσοστιαία έκφραση ως κλάσμα μεταξύ των δυο προαναφερθέντων τιμών. Τέλος στην έκτη γραμμή ορίζεται η ένδειξη της τοξοειδούς απεικόνισης και στην έβδομη το πεδίο κειμένου. Οι δύο γραμμές που εμφανίζονται μετά (i,ii) σχετίζονται με την απεικόνιση των ειδοποιήσεων.

Πέρα από την απεικόνιση των τριών αυτών μεγεθών η UpdateGUI() ανανεώνει και το πεδίο που αναφέρεται η ακριβής ώρα του τελευταίου συγχρονισμού με τον πάροχο.

6.1.7. Ειδοποιήσεις και διαχείριση αυτών.

Ο κώδικας που διαχειρίζεται τις ειδοποιήσεις είναι ίδιος με πολύ μικρές διαφοροποιήσεις για τα προπληρωμένα λεπτά ομιλίας, τα προπληρωμένα γραπτά μηνύματα και τα προπληρωμένα πακέτα διαδικτύου. Εξ αιτίας αυτού θα παρουσιαστεί εδώ μόνο ο κώδικας για τα υπολειπόμενα γραπτά μηνύματα.

```
public void ToggleMinutesNotification(View view){
      minutesToggle= !minutesToggle;
      if (minutesToggle==true){
         MinutesNotification(1);
      } else if(minutesToggle==false) {
         MinutesNotification(0);
      }
      return;
   }
```

Η μέθοδος αυτή είναι “συνδεδεμένη” με το αντίστοιχο πλήκτρο που επιτρέπει την εμφάνιση ή την απόκρυψη των ειδοποιήσεων που υπάρχει στο Γραφικό περιβάλλον της εφαρμογής. Χρησιμοποιείται κάνοντας toggle την μεταβλητή minutesToggle και καλώντας της μέθοδο MinutesNotification(). Με μηδενικό όρισμα γίνεται απόκρυψη και με όρισμα ένα γίνεται εμφάνιση της ειδοποίησης.

```
private void MinutesNotification(int minutesOption){

      Intent intent=new Intent(this,MainActivity.class);
      PendingIntent
pIntent=PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_UPDATE_CURREN
T);
      NotificationCompat.Builder builder1= new NotificationCompat.Builder(this);
      builder1.setAutoCancel(false);
      builder1.setProgress(100, minutesPercent, false);
      builder1.setContentTitle("Minutes: "+Integer.toString(minutesPercent)+"%");
      builder1.setContentText(Integer.toString(minutes)+"/"+Integer.toString(minutesMax));
      builder1.setSmallIcon(minutesIconArray[minutesPercent/10]);
      builder1.setContentIntent(pIntent);
      builder1.setOngoing(true);
      NotificationManager manager1 = (NotificationManager)
this.getSystemService(NOTIFICATION_SERVICE);
      if(minutesOption==1) {
```

```
            manager1.notify(1, builder1.build());
        } else if (minutesOption==0){
            manager1.cancel(1);
        }
    }
```

Η μέθοδος MinutesNotification() διαχειρίζεται την προβολή των ειδοποιήσεων για τα υπολειπόμενα προπληρωμένα λεπτά. Παραπάνω εξηγήθηκε πως λειτουργεί βάσει του ορίσματος που θα δεχθεί. Η ειδοποίηση ορίζεται σαν συνεχής προκειμένου να βρίσκεται διαρκώς διαθέσιμη στον χρήστη εφόσον αυτός το επιθυμεί, επίσης επιτρέπει στον χρήστη με ένα πάτημα να μεταβεί στην εφαρμογή. Τέλος, αξιοσημείωτη είναι η δυνατότητα των ειδοποιήσεων να πληροφορούν τον χρήστη με ακρίβεια 10% για το υπόλοιπό του προπληρωμένου χρόνου ομιλίας. Στην ουσία πρόκειται για μια σειρά εικονιδίων, για τα οποία οι αναφορές βρίσκονται αποθηκευμένες στον πίνακα minutesIconArray. Υπάρχουν 11 εικονίδια για τα εξής εύρη τιμών της ποσοστιαίας έκφρασης των υπολοίπων: 0%-9%, 10%-19, 20%-29, 30%-39, 40%-49, 50%-59, 60%-69, 70%-79, 80%-89, 90%-99 και 100%.

6.1.8. Ορισμός Νέου Μεγίστου.

Για τον ορισμό του νέας μεγίστης τιμής/τιμής αναφοράς για κάθε ένα από τα τρία μετρούμενα μεγέθη (προπληρωμένος χρόνος ομιλίας, προπληρωμένα γραπτά μηνύματα, προπληρωμένα πακέτα διαδικτύου) χρησιμοποιείται η κλάση SetMax. Ο ορισμός και η επεξήγηση της κλάσης φαίνεται παραπάνω. Η κλάση αυτή στην πραγματικότητα διαχειρίζεται και περιγράφει ένα fragment. Εξ ορισμού τα fragments αντιπροσωπεύουν ένα κομμάτι του γραφικού περιβάλλοντος καθώς και τις λειτουργίες που σχετίζονται με αυτό. Στην κλάση MainActivity βρίσκεται το εξής απόσπασμα κώδικα που καλείται όταν ο χρήστης πατήσει το πλήκτρο ορισμού νέου μεγίστου/τιμής αναφοράς:

```
public void SetMaxDialog (View v){
    FragmentManager manager=getFragmentManager();
    SetMax setMaxDialog=new SetMax();
    setMaxDialog.show(manager,"setMaxDialog");
    switch (v.getId()){
        case R.id.setMinutesMaxButton:
            maxSwitch=1;
```

```
            break;
        case R.id.setSMSMaxButton:
            maxSwitch=2;
            break;
        case R.id.setDataMaxButton:
            maxSwitch=3;
            break;
    }
  }
```

Εδώ αυτό που γίνεται είναι ότι η τιμή maxSwitch παίρνει την τιμή 1 αν ο χρήστης θέλει να αλλάξει την τιμή αναφοράς του προπληρωμένου χρόνου και τις τιμές 2 και 3 για τα γραπτά μηνύματα και τα πακέτα διαδικτύου αντίστοιχα.

6.1.8.1. Η κλάση SetMax.

```
public class SetMax extends DialogFragment implements View.OnClickListener {
   Button SetButton,CancelButton;
   Communicator comm;
   public void onAttach(Activity a){
      super.onAttach(a);
      comm=(Communicator)a;
   }



   @Override
   public View onCreateView(LayoutInflater inflater,ViewGroup container,Bundle
savedInstanceState){
      View view;
      view=inflater.inflate(R.layout.set_max_dialog,null);
      SetButton=(Button) view.findViewById(R.id.setButton);
      CancelButton=(Button) view.findViewById(R.id.cancelButton);
      SetButton.setOnClickListener(this);
      CancelButton.setOnClickListener(this);
      setCancelable(false);
```

```
        return view;
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        TextView tv;
        switch (MainActivity.maxSwitch){
            case 1:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
                tv.setText("SET MINUTES MAXIMUM VALUE");
                tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
                tv.setText(Integer.toString(MainActivity.minutesMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
                tv.setText("Current minutes maximum:
"+Integer.toString(MainActivity.minutesMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentValue);
                tv.setText("Current minutes value: "+Integer.toString(MainActivity.minutes));
                break;

            case 2:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
                tv.setText("SET SMS MAXIMUM VALUE");
                tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
                tv.setText(Integer.toString(MainActivity.smsMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
                tv.setText("Current SMS maximum: "+Integer.toString(MainActivity.smsMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentValue);
                tv.setText("Current SMS value: "+Integer.toString(MainActivity.sms));
                break;

            case 3:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
```

```
            tv.setText("SET DATA MAXIMUM VALUE");
            tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
            tv.setText(Integer.toString(MainActivity.dataMax));
            tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
            tv.setText("Current data maximum: "+Integer.toString(MainActivity.dataMax));
            tv = (TextView)getView().findViewById(R.id.textCurrentValue);
            tv.setText("Current data value: "+Integer.toString(MainActivity.data));
            break;
        }

    }

    @Override
    public void onClick(View v) {
        int setvalue=0;

        TextView tv;
        if(v.getId()==R.id.setButton){
            tv =(TextView) getView().findViewById(R.id.textMaximumSetter);
            if (!(tv.getText().length()==0)) {

                setvalue = Integer.parseInt(tv.getText().toString());
                if (MainActivity.maxSwitch == 1) {
                    if ((setvalue<1)||(setvalue<MainActivity.minutes)){
                        setvalue=100;
                        Toast.makeText(MainActivity.context, "Maximum must be positive and at
least equal to current value",Toast.LENGTH_LONG).show();}
                    MainActivity.minutesMax = Math.max(setvalue,MainActivity.minutes);

                } else if (MainActivity.maxSwitch == 2) {
                    if ((setvalue<1)||(setvalue<MainActivity.sms)){
                        setvalue=100;
                        Toast.makeText(MainActivity.context, "Maximum must be positive and at
```

```
least equal to current value",Toast.LENGTH_LONG).show();}
                MainActivity.smsMax = Math.max(setvalue,MainActivity.sms);



            } else {
                if ((setvalue<1)||(setvalue<MainActivity.data)){
                    setvalue=100;
                    Toast.makeText(MainActivity.context, "Maximum must be positive and at
least equal to current value",Toast.LENGTH_LONG).show();}
                MainActivity.dataMax =Math.max (setvalue,MainActivity.data);
            }
            MainActivity.maxSwitch = 0;
            dismiss();
          }else{
            dismiss();
            Toast.makeText(MainActivity.context, "wrong
form",Toast.LENGTH_LONG).show();
          }
          comm.Respond("Update");
       }
       else {
          dismiss();
       }
    }
}
```

Η κλάση βοηθητικά χρησιμοποιεί ένα Interface ονόματι communicator το οποίο αναλύεται παρακάτω.

Με την μέθοδο onAttach συσχετίζεται η εκτέλεση του interface αυτού με τον κώδικα που περιγράφει την λειτουργία του στο MainActivity.

Η μέθοδος onCreateView εκτελείται με το που εμφανίζεται το fragment στην οθόνη. Η μέθοδος εμφανίζει το κομμάτι του γραφικού περιβάλλοντος στην οθόνη, συσχετίζει τα δυο πλήκτρα με τις μεταβλητές και επιτρέπει την έξοδο από το γραφικό περιβάλλον μόνο με την

χρήση των πλήκτρων.

Η μέθοδος onViewCreated συμπληρώνει τα πεδία με τις τιμές που αντιστοιχούν στο μέγεθος του οποίου θα γίνει ορισμός νέας τιμής αναφοράς.

Η μέθοδος onClick περιγράφει την συμπεριφορά της εφαρμογής στο πάτημα των δύο πλήκτρων. Με το ένα πλήκτρο ορίζεται η νέα τιμή αναφοράς και επιστροφή στην κεντρική όψη της εφαρμογής και με το άλλο γίνεται απλά επιστροφή στην κεντρική όψη. Επιπλέον γίνεται έλεγχος προκειμένου να τηρούνται όλες οι προϋποθέσεις ακεραιότητας για τα μετρούμενα μεγέθη.

6.1.8.2. Το interface Communicator.

```
interface Communicator{
   public void Respond(String data);
}
```

Το interface είναι πάρα πολύ απλό με μια μόνο μέθοδο το σώμα της οποίας περιγράφεται στην MainActivity και είναι το ακόλουθο.

```
public void Respond(String data){
      UpdateGUI();
   }
```

Πρακτικά αυτό σημαίνει ότι με το που θα γίνει ορισμός νέου μεγίστου, αυτόματα θα υπάρξει ανανέωση του γραφικού περιβάλλοντος.

6.1.9. Διασταύρωση Δεδομένων με τον Πάροχο.

Για την διασταύρωση των δεδομένων (προπληρωμένος χρόνος ομιλίας, προπληρωμένα γραπτά μηνύματα, προπληρωμένα πακέτα διαδικτύου) με τον πάροχο η εφαρμογή χρειάζεται να μπορεί να πραγματοποιήσει τις εξής δύο λειτουργίες. Πρώτον, να μπορεί να αποστείλει γραπτά μηνύματα και δεύτερον να μπορεί να εξετάσει τα εισερχόμενα μηνύματα για το αν προέρχονται από τον πάροχο και για το αν είναι μηνύματα που σχετίζονται με την εξυπηρέτηση της εφαρμογής.

Για αυτόν τον λόγο έχουμε γράψει τις ακόλουθες μεθόδους στην mainActivity αλλά και την κλάση SMSReceiver.

6.1.9.1. Δύο μέθοδοι της κλάσης MainActivity

```
public void SMSClick(View view){
        sendSMS("XXXXXXXXX", "TestTest");
        Toast.makeText(getApplicationContext(), "Sending SMS",
Toast.LENGTH_SHORT).show();
        return;
    }
```

Η μέθοδος αυτή καλείται όταν ο χρήστης επιθυμεί να εκκινήσει χειροκίνητα την διαδικασία συγχρονισμού με τον πάροχο, πατώντας το ανάλογο πλήκτρο. Στην ουσία πρόκειται για μια κλήση στην μέθοδο sendSMS (που εξηγείται παρακάτω) με ορίσμα τον αριθμό του παρόχου (αντιπροσωπευόμενος εδώ με τον αριθμό XXXXXXX) και το περιεχόμενο του μηνύματος (αντιπροσωπευόμενο εδώ με το κείμενο TestTest). Επίσης εμφανίζει ένα μικρό μήνυμα (toast) στην οθόνη με το μήνυμα Sending Sms.

```
private void sendSMS(String phoneNumber, String message)
    {
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, null, null);
    }
```

Η μέθοδος sendSMS στέλνει ένα γραπτό μήνυμα με αριθμό παραλήπτη και περιεχόμενο όπως ορίζονται από τα ορίσματά της.

6.1.9.2. Η κλάση SMSReceiver.

```
package kozos.trindicator;


import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
import java.util.Date;
```

```
public class SMSReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent)
   {
      String[] strArray;
      Bundle bundle = intent.getExtras();
      SmsMessage[] msgs = null;
      String str = "";
      if (bundle != null) {
         Object[] pdus = (Object[]) bundle.get("pdus");
         msgs = new SmsMessage[pdus.length];
         for (int i = 0; i < msgs.length; i++) {
            msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
            str += msgs[i].getMessageBody().toString();
            Toast.makeText(context, msgs[0].getOriginatingAddress().toString() + "  " + str,
Toast.LENGTH_SHORT).show();
            strArray = str.split(" ");
            if (((msgs[0].getOriginatingAddress().equals("XXXXXXXXXX") ||
msgs[0].getOriginatingAddress().equals("+30XXXXXXXXXX"))) &&
strArray[0].equals("0")) {
               MainActivity.minutes = Integer.parseInt(strArray[1]);

MainActivity.minutesMax=Math.max(MainActivity.minutes,MainActivity.minutesMax);
               MainActivity.sms = Integer.parseInt(strArray[2]);
               MainActivity.smsMax=Math.max(MainActivity.sms,MainActivity.smsMax);
               MainActivity.data = Integer.parseInt(strArray[3]);
               MainActivity.dataMax=Math.max(MainActivity.data,MainActivity.dataMax);
               MainActivity.lastUpdate=new Date();

MainActivity.UpdateDate=MainActivity.formatter.format(MainActivity.lastUpdate);
               MainActivity.hasUpdated=true;
```

```
                MainActivity.previousTotalBytes=MainActivity.res;
                MainActivity.lastUpdateMS=System.currentTimeMillis();
                }
            }
        }
    }
```

Η κλάση SMSReceiver είναι ένας Broadcast Receiver που ενεργοποιείται όταν λαμβάνεται γραπτό μήνυμα. Θεωρείται ότι το γραπτό μήνυμα με το οποίο γίνεται ο συγχρονισμός είναι ένα γραπτό μήνυμα τεσσάρων αριθμών, με τον πρώτο να είναι 0 τον επόμενο να είναι το υπόλοιπο των προπληρωμένων λεπτών τον τρίτο αριθμό να είναι το υπόλοιπο των προπληρωμένων γραπτών μηνυμάτων και τον τελευταίο να είναι τα υπολοιπόμενα πακέτα διαδικτύου σε Mbyte. Γίνεται έλεγχος για το αν το γραπτό μήνυμα είναι από τον αριθμό του παρόχου και αν ο πρώτος αριθμός είναι το μηδέν. Αν ισχύουν αυτές οι προϋποθέσεις τότε γίνεται ανανέωση των στοιχείων. Σε αντίθετη περίπτωση το μήνυμα αγνοείται.

6.1.10. Το παράθυρο βοήθειας.

Στην κεντρική όψη της εφαμοργής υπάρχει πλήκτρο το οποίο δίνει την δυνατότητα στον χρήστη να δει σε αναδυόμενο παράθυρο οδηγίες για το πώς να χρησιμοποιήσει την εφαρμογή.

6.1.10.1. Η μέθοδος HelpDialog της κλάσης MainActivity.

```
public void HelpDialog(View v){
    FragmentManager manager=getFragmentManager();
    HelpDialog helpDialog=new HelpDialog();
    helpDialog.show(manager,"helpDialog");
  }
```

Η μέθοδος αυτή εκτελείται με το πάτημα του πλήκτρου Help και εμφανίζει το αναδυόμενο παράθυρο.

6.1.10.2. Η κλάση HelpDialog.

```
package kozos.trindicator;
```

```
import android.app.DialogFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

public class HelpDialog extends DialogFragment implements View.OnClickListener {
   Button OkButton;

   public View onCreateView(LayoutInflater inflater,ViewGroup container,Bundle
savedInstanceState){
      View view;
      view=inflater.inflate(R.layout.help_dialog,null);
      OkButton=(Button) view.findViewById(R.id.OkHelpButton);
      OkButton.setOnClickListener(this);
      setCancelable(false);
      return view;
   }

   public void onClick(View v){
      dismiss();
   }
}
```

Η κλάση αυτή ελέγχει την εμφάνιση του αναδυόμενου παραθύρου βοηθείας και την συμπεριφορά του. Η εμφάνιση γίνεται με την εντολή inflate και ορίζουμε πλήκτρο, μόνο μέσω του οποίου είναι δυνατόν να κλεισθεί το παράθυρο.

6.2. Το γραφικό περιβάλλον.

Σε μία εφαρμογή android τα στοιχεία που αποτελούν το γραφικό περιβάλλον μπορούν να δημιουργηθούν με δύο τρόπους. Είτε με κώδικα java που επιτρέπει την δυναμική δημιουργία τους είτε με στατικό κώδικα xml. Το γραφικό περιβάλλον της εφαρμογής έχει γραφεί σε μεγάλο βαθμό σε κώδικα xml με εξαίρεση την ανοικτή βιβλιοθήκη που χρησιμοποιήθηκε για

τις τοξοειδείς απεικονίσεις, η οποία είναι γραμμένη σε java. Κατά την σχεδίαση του γραφικού περιβάλλοντος δόθηκε ιδιαίτερη έμφαση ώστε να είναι όσο το δυνατόν πιο περιεκτικό και φιλικό προς τον χρήστη αλλά ταυτόχρονα να είναι εύκολη η πρόσβαση στις πληροφορίες που εμφανίζει η εφαρμογή. Οι xml κώδικες που αποτελούν το γραφικό περιβάλλον βρίσκονται στο Παράρτημα Α.

6.2.1. Η ελεύθερη βιβλιοθήκη.

Για τους τοξοειδείς ενδείκτες που χρησιμοποιούνται στο γραφικό περιβάλλον έγινε χρήση της ελεύθερης βιβλιοθήκης CircleProgress. Η βιβλιοθήκη αυτή προσφέρει στον προγραμματιστή την δυνατότητα να εμφανίσει και να χειριστεί τοξοειδείς, σφαιρικούς και τοροειδείς ενδείκτες. Η βιβλιοθήκη παρέχεται ανοικτά και επιτρέπεται η χρήση ή η επεξεργασία της για κάθε είδους σκοπό. Στην Βιβλιογραφία παρέχεται σύνδεσμος προς αυτήν.

6.3. Το manifest.

Κάθε εφαρμογή android πρέπει να περιλαμβάνει ένα αρχείο xml με το όνομα AndroidManifest.xml. Το αρχείο αυτό περιλαμβάνει απαραίτητες πληροφορίες για την λειτουργία και την εκτέλεση της εφαρμογής από το λειτουργικό σύστημα, πληροφορίες που χρειάζεται να είναι γνωστές πριν την εκτέλεση της εφαρμογής. Πολύ συνοπτικά μπορούμε να πούμε ότι το manifest περιλαμβάνει μια λίστα με τις ανάγκες τις εφαρμογής για την ορθή εκτέλεσή της καθώς και μια συνοπτική παρουσίαση των επιμέρους κομματιών της και πως αυτά αλληλεπιδρούν, τόσο μεταξύ τους όσο και με τις δυνατότητες που προσφέρει το λειτουργικό σύστημα.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="kozos.trindicator" >

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

```
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:launchMode="singleInstance" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".PhoneStateBroadcastReceiver">
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE">
                </action></intent-filter>
        </receiver>
        <receiver android:name=".SMSReceiver">
            <intent-filter>
                <action android:name=
                    "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
        <receiver android:name=".SMSOutgoing">
            <intent-filter>
                <action android:name=
                    "android.provider.Telephony.SMS_SENT" />
            </intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.READ_PHONE_STATE">
    </uses-permission>
    <uses-permission android:name="android.permission.READ_SMS"></uses-permission>
    <uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
```

```
    <uses-permission android:name="android.permission.RECEIVE_SMS">
    </uses-permission>
    <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

Στο παραπάνω manifest αρχικά ορίζεται το όνομα του package, μέρος του οποίου είναι και η εφαρμογή. Μετά ακολουθούν κάποιοι ορισμοί για την εφαρμογή. Ο πλέον αξιοσημείωτος είναι ο εξής:

```
android:launchMode="singleInstance"
```

Ορίζοντας την εντολή σαν single instance δεν επιτρέπεται η ταυτόχρονη λειτουργία πολλαπλών στιγμιοτύπων της εφαρμογής, ανάγκη που εξηγείται παραπάνω στην Παράγραφο 5.3 (Ακρίβεια σε πολλαπλές ενεργοποιήσεις της εφαρμογής).

Μετά συσχετίζονται οι Broadcast Receivers με τα events που πρέπει να παρακολουθούν.

Event	Broadcast Receiver
PhoneState	PhoneStateBroadcastReceiver
SMS_RECEIVED	SMSReceiver
SMS_SENT	SMSOutgoing

Τέλος ορίζονται οι άδειες (permissions) που χρειάζεται η εφαρμογή για να εκτελεστεί. Στο λειτουργικό android, προκειμένου να προστατευθεί ο χρήστης από κακόβουλο λογισμικό, κάθε εφαρμογή πρέπει να δηλώνει εξ αρχής σε ποια ευαίσθητα στοιχεία του χρήστη θέλει να έχει πρόσβαση. Ο χρήστης ενημερώνεται για αυτά κατά την εγκατάσταση της εφαρμογής και αν δεν αποδέχεται την χρήση αυτών μπορεί να σταματήσει την εγκατάσταση. Στην εφαρμογή γίνεται χρήση των ακόλουθων αδειών:

Read Phone State

Read SMS

Process Outgoing Calls

Receive SMS

Send SMS.

Κεφάλαιο 7

Προτάσεις Βελτίωσης.

7.1 Προτάσεις για μελλοντική βελτίωση.

Από την εμπειρία που αποκτήθηκε κατά την εκπόνηση της παρούσας εργασίας γίνεται φανερό ότι ιδανικά μια τέτοιου είδους εφαρμογή θα πρέπει να μπορεί να λειτουργήσει σε πραγματικό περιβάλλον, χωρίς να προκαλεί κάποια περαιτέρω επιβάρυνση στα δίκτυα του παρόχου και έχοντας διαρκώς μηδενικό σφάλμα σε όλα τα μετρούμενα μεγέθη. Για την πραγματοποίηση κάτι τέτοιου θα ήταν αναγκαίο να γίνουν κάποιες παραδοχές. Ενδεικτικά παρατίθενται παρακάτω οι παραδοχές που θα χρειαζόταν να ισχύουν προκειμένου η εφαρμογή trindicator, με κάποιες απλές τροποποιήσεις, να μπορούσε να λειτουργήσει έτσι :

i. Παραδοχές για τις κλήσεις:

- Όλες οι κλήσεις ακολουθούν το ίδιο μοντέλο χρέωσης ανεξάρτητα του κληθέντος αριθμού ή τα προπληρωμένα λεπτά μπορούν να χρησιμοποιηθούν για κλήσεις ανεξαρτήτως του μοντέλου χρέωσης που θα ίσχυε αν δεν ήταν διαθέσιμα.
- Η ελάχιστη χρήση των προπληρωμένων λεπτών είναι ένα λεπτό της ώρας και η χρήση τους γίνεται ανά λεπτό της ώρας.
- Δεν υπάρχει καμία απόκλιση μεταξύ του υπολογισμού της διάρκειας των κλίσεων μεταξύ συσκευής και τηλεφωνικού κέντρου.

ii. Παραδοχές για τα γραπτά μηνύματα:

- Όλα τα γραπτά μηνύματα ακολουθούν το ίδιο μοντέλο χρέωσης ανεξάρτητα από τον αποστολέα ή τον παραλήπτη ή τα προπληρωμένα μηνύματα μπορούν να χρησιμοποιηθούν ανεξάρτητα του μοντέλου χρέωσης που θα ίσχυε αν δεν ήταν διαθέσιμα.
- Η χρέωση γίνεται ανά γραπτό μήνυμα.

iii. Παραδοχές για τα δεδομένα διαδικτύου:

- Η εφαρμογή θα έχει εκκινήσει και δεν θα έχει τερματιστεί μέχρι την απενεργοποίηση της συσκευής εφόσον έχει γίνει χρήση προπληρωμένων δεδομένων διαδικτύου.
- Δεν υπάρχει καμία απώλεια πακέτων με προορισμό την συσκευή ή, εναλλακτικά, δεν υπάρχει χρέωση για αυτά τα χαμένα πακέτα.

iv. Παραδοχές για την λειτουργία της εφαρμογής από τον χρήστη:

- Ο χρήστης θα ενεργοποιεί την εφαρμογή με το που θα ενεργοποιήσει την συσκευή.
- Ο χρήστης δεν θα απενεργοποιεί ποτέ την εφαρμογή όσο η συσκευή βρίσκεται σε λειτουργία.

  Σημείωση: Οι παρούσες παραδοχές λαμβάνουν υπόψιν τους περιορισμούς του λειτουργικού συστήματος Android που σε μεγάλο βαθμό κληρονομήθηκαν από τα τηλεπικοινωνιακά συστήματα.

Γίνεται εύκολα αντιληπτό ότι όλες αυτές οι παραδοχές θα μπορούσαν να ικανοποιούνται διαρκώς μόνο σε ένα εξιδανικευμένο τηλεπικοινωνιακό περιβάλλον. Παρ όλα αυτά από τις παραδοχές μπορεί να εξαχθεί ένα πολύ χρήσιμο συμπέρασμα, ότι μια εφαρμογή τέτοιου είδους καλείται να συμβιβάσει δυο αμοιβαίως αποκλειόμενους στόχους: μηδενική επιβάρυνση του παρόχου και μηδενικές αποκλίσεις σε όλα τα μετρούμενα μεγέθη. Το ασύμβατο όμως των δύο αυτών στόχων σημαίνει ότι ο συμβιβασμός είναι αναγκαίος και η ποιότητα της εφαρμογής έγκειται στο πόσο αποτελεσματικά μπορεί να ικανοποιήσει τον ένα χωρίς να υπονομεύσει τον άλλον. Συγκεκριμένα στην παρούσα εφαρμογή γίνεται αυτόματη ενημέρωση του χρήστη ανά δύο ώρες με γραπτό μήνυμα και επίσης παρέχεται η δυνατότητα να αποστείλει αίτημα ανανέωσης όποτε το επιθυμεί ο ίδιος. Αυτή η αποστολή μηνυμάτων ανά δύο ώρες προς όλους τους χρήστες αποτελεί σοβαρή επιβάρυνση των υποδομών του παρόχου. Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί το διαδίκτυο, να γίνεται δηλαδή μέσω αυτού η ανανέωση της εφαρμογής, αντί για γραπτό μήνυμα. Κάτι τέτοιο, πέρα από την σημαντική ελάφρυνση του δικτύου του παρόχου, θα ήρε την ανάγκη για τις προαναφερθείσες παραδοχές και θα επέτρεπε στην εφαρμογή να παρακολουθεί πιο εξειδικευμένες προσφορές που τοπικά, λόγω των περιορισμών του λειτουργικού συστήματος, είναι αδύνατο να μετρηθούν.

Παράρτημα Α.

Ο Πλήρης Κώδικας της εφαρμογής.

Α.1 Η δομή της εφαρμογής. (Directories)

App/manifests/AndroidManifest.xml

/java/kozos.trindicator/HelpDialog.java
/MainActivity.java
/PhoneStateBroadcasterReceiver.java
/SentSMSObserver.java
/SetMax.java
/Communicator.java
/SMSReceiver.java

/res/drawable/data0.png
/data10.png
/data100.png
/data20.png
/data30.png
/data40.png
/data50.png
/data60.png
/data70.png
/data80.png
/data90.png
/minutes0.png
/minutes10.png
/minutes100.png
/minutes20.png
/minutes30.png
/minutes40.png
/minutes50.png
/minutes60.png

/minutes70.png
/minutes80.png
/minutes90.png
/sms0.png
/sms10.png
/sms100.png
/sms20.png
/sms30.png
/sms40.png
/sms50.png
/sms60.png
/sms70.png
/sms80.png
/sms90.png
/trindicatoricon.png

/res/layout/activity_main.xml
/help_dialog.xml
/set_max_dialog.xml

A.2. Τα αρχεία .java (στο directory: App/java/kozos.trindicator/)

A.2.1. HelpDialog.java

```
package kozos.trindicator;

import android.app.DialogFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

public class HelpDialog extends DialogFragment implements View.OnClickListener {
```

```
    Button OkButton;

    public View onCreateView(LayoutInflater inflater,ViewGroup container,Bundle
savedInstanceState){
        View view;
        view=inflater.inflate(R.layout.help_dialog,null);
        OkButton=(Button) view.findViewById(R.id.OkHelpButton);
        OkButton.setOnClickListener(this);
        setCancelable(false);
        return view;
    }

    public void onClick(View v){
        dismiss();
    }
}
```

A.2.2. MainActivity.java

```
package kozos.trindicator;

import android.app.Activity;
import android.app.FragmentManager;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.TrafficStats;
import android.net.Uri;
import android.os.Handler;
import android.os.SystemClock;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.ActionBarActivity;
```

```
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import com.github.lzyzsd.circleprogress.ArcProgress;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainActivity extends ActionBarActivity implements Communicator{
    public static int
minutes=55,minutesMax=100,minutesPercent=55,sms=25,smsMax=100,smsPercent=25,dat
a=85,dataMax=100,dataPercent=85,maxSwitch=0;
    public static boolean
minutesToggle=false,smsToggle=false,dataToggle=false,hasUpdated=false;
    public static String UpdateDate="";
    public static String lastMessage = "-a";
    public static String thisMessage = null;
    public int whoCalledMax=0;
    public static long res=0L;
    int reducer=0;
    public static long previousTotalBytes;
    long lastDataCheck;
    long bootTime;
    public static Activity activityReference;
    static long lastUpdateMS;
    static Date lastUpdate;
    SentSMSObserver myObserver;
    public static SimpleDateFormat formatter=new SimpleDateFormat("HH:mm
dd/MM/yyyy");
    public static Context context;
    public static Activity activity;
    TrafficStats MyTrafficStats=new TrafficStats();
```

```
   int[] minutesIconArray =
{R.drawable.minutes0,R.drawable.minutes10,R.drawable.minutes20,R.drawable.minutes30,
R.drawable.minutes40,R.drawable.minutes50,

R.drawable.minutes60,R.drawable.minutes70,R.drawable.minutes80,R.drawable.minutes90
,R.drawable.minutes100};
   int[]
smsIconArray={R.drawable.sms0,R.drawable.sms10,R.drawable.sms20,R.drawable.sms30,
R.drawable.sms40,R.drawable.sms50,R.drawable.sms60,R.drawable.sms70,
         R.drawable.sms80,R.drawable.sms90,R.drawable.sms100};
   int[]
dataIconArray={R.drawable.data0,R.drawable.data10,R.drawable.data20,R.drawable.data3
0,R.drawable.data40,R.drawable.data50,R.drawable.data60,
         R.drawable.data70,R.drawable.data80,R.drawable.data90,R.drawable.data100};
   public static ContentResolver contentResolver;
   public void Respond(String data){
      UpdateGUI();
   }

   @Override
   public void onResume() {
      super.onResume();
      handlerStoreAndUpdate.removeCallbacks(runnableCode1);
      handlerDataAndGUI.removeCallbacks(runnableCode2);
      UpdateGUI();
      activityReference=this;
      handlerStoreAndUpdate.post(runnableCode1);
      handlerDataAndGUI.post(runnableCode2);
   }
```

```
@Override
protected void onDestroy(){
    super.onDestroy();
    Store();
    minutesToggle=false;
    MinutesNotification(0);
    smsToggle=false;
    SMSNotification(0);
    dataToggle=false;
    DataNotification(0);
    getContentResolver().unregisterContentObserver(myObserver);
}

void Store(){
    SharedPreferences sharedPreferences= getPreferences(MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt("minutes",minutes);
    editor.putLong("lastDataCheck",lastDataCheck);
    editor.putInt("minutesMax",minutesMax);
    editor.putInt("sms",sms);
    editor.putInt("smsMax",smsMax);
    editor.putInt("data",data);
    editor.putInt("dataMax",dataMax);
    editor.putBoolean("hasUpdated",hasUpdated);
    editor.putLong("previousTotalBytes",previousTotalBytes);
    editor.putLong("lastUpdateMS",lastUpdateMS);
    editor.putString("MainActivitylastMessage",lastMessage);
    editor.putString("MainActivitythisMessage",thisMessage);
    if(hasUpdated==true){
        editor.putString("UpdateDate",UpdateDate);
    }
    editor.commit();
}
```

```
public void UpdateGUI(){

    ArcProgress ap = (ArcProgress) findViewById(R.id.arcMinutes);
    TextView tv=(TextView) findViewById(R.id.textMinutesDisplay);
    minutes=Math.max(minutes,0);
    minutesMax=Math.max(Math.max(minutes,minutesMax),100);
    minutesPercent=(100*minutes)/minutesMax;
    ap.setProgress(minutesPercent);
    tv.setText(" " + Integer.toString(minutes) + " / " + Integer.toString(minutesMax) + " ");
    ap = (ArcProgress) findViewById(R.id.arcSMS);
    tv=(TextView) findViewById(R.id.textSMSDisplay);
    sms=Math.max(sms,0);
    smsMax=Math.max(Math.max(sms,smsMax),100);
    smsPercent=(100*sms)/smsMax;
    ap.setProgress(smsPercent);
    tv.setText(" " + Integer.toString(sms) + " / " + Integer.toString(smsMax) + " ");
    ap = (ArcProgress) findViewById(R.id.arcData);
    tv=(TextView) findViewById(R.id.textDataDisplay);
    data=Math.max(data,0);
    dataMax=Math.max(Math.max(data,dataMax),100);
    dataPercent=(100*data)/dataMax;
    ap.setProgress(dataPercent);
    tv.setText(" "+Integer.toString(data)+" / "+Integer.toString(dataMax)+" ");
    if (hasUpdated==true){
        tv=(TextView) findViewById(R.id.textLastUpdate);
        tv.setText(" Last Update: "+UpdateDate+" ");
    }else{
        tv=(TextView) findViewById(R.id.textLastUpdate);
        tv.setText("  PENDING  ");
    }
    if (minutesToggle==true){
        MinutesNotification(1);
```

```
        }
        if (smsToggle==true){
            SMSNotification(1);
        }
        if (dataToggle==true){
            DataNotification(1);
        }
    }

    public void ToggleMinutesNotification(View view){
        minutesToggle= !minutesToggle;
        if (minutesToggle==true){
            MinutesNotification(1);
        } else if(minutesToggle==false) {
            MinutesNotification(0);
        }
        return;
    }
    private void MinutesNotification(int minutesOption){

        Intent intent=new Intent(this,MainActivity.class);
        PendingIntent
pIntent=PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_UPDATE_CURREN
T);
        NotificationCompat.Builder builder1= new NotificationCompat.Builder(this);
        builder1.setAutoCancel(false);
        builder1.setProgress(100, minutesPercent, false);
        builder1.setContentTitle("Minutes: "+Integer.toString(minutesPercent)+"%");
        builder1.setContentText(Integer.toString(minutes)+"/"+Integer.toString(minutesMax));
        builder1.setSmallIcon(minutesIconArray[minutesPercent/10]);
        builder1.setContentIntent(pIntent);
        builder1.setOngoing(true);
        NotificationManager manager1 = (NotificationManager)
```

```
this.getSystemService(NOTIFICATION_SERVICE);
        if(minutesOption==1) {
            manager1.notify(1, builder1.build());
        } else if (minutesOption==0){
            manager1.cancel(1);
        }
    }
    public void ToggleSMSNotification(View view){
        smsToggle= !smsToggle;
        if (smsToggle==true){
            SMSNotification(1);
        } else if (smsToggle==false){
            SMSNotification(0);
        }
        return;
    }
    private void SMSNotification(int smsOption){
        Intent intent=new Intent(this,MainActivity.class);
        PendingIntent
pIntent=PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_UPDATE_CURREN
T);
        NotificationCompat.Builder builder2= new NotificationCompat.Builder(this);
        builder2.setAutoCancel(false);
        builder2.setProgress(100, smsPercent, false);
        builder2.setContentTitle("SMS: "+Integer.toString(smsPercent)+"%");
        builder2.setContentText(Integer.toString(sms)+"/"+Integer.toString(smsMax));
        builder2.setSmallIcon(smsIconArray[smsPercent/10]);
        builder2.setContentIntent(pIntent);
        builder2.setOngoing(true);
        NotificationManager manager2 = (NotificationManager)
this.getSystemService(NOTIFICATION_SERVICE);
        if(smsOption==1) {
            manager2.notify(2, builder2.build());
```

```
        } else if (smsOption==0){
            manager2.cancel(2);
        }
    }
    public void ToggleDataNotification(View view) {
        dataToggle = !dataToggle;
        if (dataToggle == true) {
            DataNotification(1);
        } else if (dataToggle == false) {
            DataNotification(0);
            return;
        }
    }
    private void DataNotification(int dataOption){
        Intent intent=new Intent(this,MainActivity.class);
        PendingIntent
pIntent=PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_UPDATE_CURREN
T);
        NotificationCompat.Builder builder3= new NotificationCompat.Builder(this);
        builder3.setAutoCancel(false);
        builder3.setProgress(100, dataPercent, false);
        builder3.setContentTitle("Data: "+Integer.toString(dataPercent)+"%");
        builder3.setContentText(Integer.toString(data)+"/"+Integer.toString(dataMax));
        builder3.setSmallIcon(dataIconArray[dataPercent/10]);
        builder3.setContentIntent(pIntent);
        builder3.setOngoing(true);
        NotificationManager manager3 = (NotificationManager)
this.getSystemService(NOTIFICATION_SERVICE);
        if(dataOption==1) {
            manager3.notify(3, builder3.build());
        } else if (dataOption==0){
            manager3.cancel(3);
        }
```

```
}
Handler handlerStoreAndUpdate = new Handler();

private Runnable runnableCode1 = new Runnable() {
   @Override
   public void run() {
      Store();
      if((hasUpdated)&&(System.currentTimeMillis()-lastUpdateMS>2*3600000)){
         sendSMS("XXXXXXXXXX", "TestTest");
      }
      handlerStoreAndUpdate.postDelayed(runnableCode1, 10*60000);
   }
};

Handler handlerDataAndGUI = new Handler();
private Runnable runnableCode2 = new Runnable() {
   @Override
   public void run() {
   res=MyTrafficStats.getMobileTxBytes();
      lastDataCheck=System.currentTimeMillis();
   if (res-previousTotalBytes>1000000L-1L){
      reducer=(int)((res-previousTotalBytes)/1000000L);
      data=data-reducer;
      data=Math.max(data,0);
      previousTotalBytes=previousTotalBytes+(long)reducer*1000000L;
      UpdateGUI();
   }
      handlerDataAndGUI.postDelayed(runnableCode2, 1*60000);
   }
};

public void HelpDialog(View v){
   FragmentManager manager=getFragmentManager();
```

```
    HelpDialog helpDialog=new HelpDialog();
    helpDialog.show(manager,"helpDialog");
}

public void SetMaxDialog (View v){

    FragmentManager manager=getFragmentManager();
    SetMax setMaxDialog=new SetMax();
    setMaxDialog.show(manager,"setMaxDialog");
    switch (v.getId()){

        case R.id.setMinutesMaxButton:
            maxSwitch=1;

            break;
        case R.id.setSMSMaxButton:
            maxSwitch=2;
            break;
        case R.id.setDataMaxButton:
            maxSwitch=3;
            break;
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    bootTime=System.currentTimeMillis()- SystemClock.elapsedRealtime();
    activity=this;
    setContentView(R.layout.activity_main);
    context=getApplicationContext();
    SharedPreferences sharedPreferences= getPreferences(MODE_PRIVATE);
    minutes=sharedPreferences.getInt("minutes",0);
```

```
        minutesMax=sharedPreferences.getInt("minutesMax",100);
        sms=sharedPreferences.getInt("sms",0);
        smsMax=sharedPreferences.getInt("smsMax",100);
        data=sharedPreferences.getInt("data",0);
        dataMax=sharedPreferences.getInt("dataMax",100);
        hasUpdated=sharedPreferences.getBoolean("hasUpdated", false);
        lastMessage=sharedPreferences.getString("MainActivitylastMessage", "-a");
        thisMessage=sharedPreferences.getString("MainActivitythisMessage",null);
        lastDataCheck=sharedPreferences.getLong("lastDataCheck",0);
        lastUpdateMS=sharedPreferences.getLong("lastUpdateMS",0);

previousTotalBytes=sharedPreferences.getLong("previousTotalBytes",MyTrafficStats.getM
obileTxBytes());
        if (bootTime>lastDataCheck)
        {
           previousTotalBytes=0;
        }
        if (hasUpdated==true) {
           UpdateDate=sharedPreferences.getString("UpdateDate","");
        }
        Handler handle = new Handler(){};
        myObserver = new SentSMSObserver(handle);
        if (hasUpdated==true) {
           myObserver.CheckPast();
        }
        contentResolver = getContentResolver();
        contentResolver.registerContentObserver(Uri.parse("content://sms"),true,
myObserver);
        activityReference=this;

    }

    public void SMSClick(View view){
```

```
        sendSMS("XXXXXXXXXX", "TestTest");
        Toast.makeText(getApplicationContext(), "Sending SMS",
Toast.LENGTH_SHORT).show();
        return;
    }

    private void sendSMS(String phoneNumber, String message)
    {
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, null, null);
    }
}
```

A.2.3. PhoneStateBroadcastReceiver.java

```
package kozos.trindicator;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import java.util.Calendar;

public class PhoneStateBroadcastReceiver extends BroadcastReceiver {

    private final String TAG = "PhoneState";
    Communicator comm2;
    static Context mContext;
    String incoming_number;
    private static int prev_state;
    static int isOn=0;
    public String phoneNumber="1";
    public String ID;
```

```
    public static Calendar callEnd,callStart;
    long callDuration;

    @Override
    public void onReceive(Context context, Intent intent) {
        mContext = context;
        Intent startIntent = new Intent(mContext, MainActivity.class);
        startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(startIntent);
        comm2=(Communicator)MainActivity.activity;
        TelephonyManager telephony =
(TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
        CustomPhoneStateListener customPhoneListener = new CustomPhoneStateListener();
        telephony.listen(customPhoneListener, PhoneStateListener.LISTEN_CALL_STATE);
        ID= telephony.getSimOperator();
        Bundle bundle = intent.getExtras();
        phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
        String phoneNr = bundle.getString("incoming_number");
    }
    public class CustomPhoneStateListener extends PhoneStateListener {

        private final String TAG = "PhoneStateListener";

        @Override
        public void onCallStateChanged(int state, String incomingNumber){

            if( incomingNumber != null && incomingNumber.length() > 0 )
                incoming_number = incomingNumber;

            switch(state){
                case TelephonyManager.CALL_STATE_RINGING:
                    prev_state=state;
                    break;
```

```
            case TelephonyManager.CALL_STATE_OFFHOOK:
               if ((isOn==0)&&(prev_state!=TelephonyManager.CALL_STATE_RINGING))
{
                  isOn=1;
                  prev_state=state;
                  callStart= Calendar.getInstance();
               }

               break;
            case TelephonyManager.CALL_STATE_IDLE:
               if((prev_state == TelephonyManager.CALL_STATE_OFFHOOK)){
               if (isOn==1){
                  isOn=0;
                  prev_state=state;
                  callEnd= Calendar.getInstance();
                  callDuration= (callEnd.getTimeInMillis()-callStart.getTimeInMillis())/
(1000);
                  callDuration=1+callDuration/60;
                  if (callDuration>=MainActivity.minutes){
                     MainActivity.minutes=0;
                  }
                  MainActivity.minutes= (int) (MainActivity.minutes-callDuration);
                  MainActivity.minutesPercent=
(MainActivity.minutes*100/MainActivity.minutesMax);
                  Intent startIntent = new Intent(mContext, MainActivity.class);
                  startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                  mContext.startActivity(startIntent);
               }
               }
               if((prev_state == TelephonyManager.CALL_STATE_RINGING)){
                  prev_state=state;
               }
```

```
                break;
            }
        }
    }
}
```

A.2.4. SentSMSObserver.java

```
package kozos.trindicator;

import android.database.ContentObserver;
import android.database.Cursor;
import android.database.CursorIndexOutOfBoundsException;
import android.net.Uri;
import android.os.Handler;
import android.util.Log;
import android.widget.Toast;

public class SentSMSObserver extends ContentObserver{

    public SentSMSObserver(Handler handler) {
        super(handler);
    }
    @Override
    public boolean deliverSelfNotifications(){
        return true;
    }
    public void CheckPast(){
        int i=0;
        String pastType,pastId;
        Uri smsuri = Uri.parse("content://sms/sent");
        Cursor cursor = MainActivity.context.getContentResolver().query(smsuri, null, null,
null, null);
        if (!(MainActivity.lastMessage.equals("-
a")&&(MainActivity.thisMessage.equals(null)))) {
```

```
            cursor.moveToFirst();
            while(!
(cursor.getString(cursor.getColumnIndex("_id")).equals(MainActivity.lastMessage))) {
                if (cursor.getString(cursor.getColumnIndex("type")).equals("2")){
                i++;
                }
            cursor.moveToNext();
            }
        cursor.moveToFirst();
        MainActivity.lastMessage = cursor.getString(cursor.getColumnIndex("_id"));
        MainActivity.sms=MainActivity.sms-i;
        MainActivity.sms=Math.max(MainActivity.sms,0);
        MainActivity.smsPercent=(100*MainActivity.sms)/MainActivity.smsMax;
        if ((MainActivity.sms<=10)&&(MainActivity.sms>1)){
            Toast.makeText(MainActivity.context, "Running Out Of free SMS",
Toast.LENGTH_LONG).show();
        }
        if(MainActivity.sms==0){
            Toast.makeText(MainActivity.context, "You have run out of Free
SMS",Toast.LENGTH_LONG).show();
        }
        }
    }
    @Override
    public void onChange(boolean selfChange){
        super.onChange(selfChange);
        Uri smsuri = Uri.parse("content://sms/sent");
        Cursor cursor = MainActivity.context.getContentResolver().query(smsuri, null, null,
null, null);
        String outgoingSMS = null;
        if (cursor != null && cursor.moveToFirst()){
            try{
            cursor.moveToFirst();
```

```
            String type = cursor.getString(cursor.getColumnIndex("type"));
            MainActivity.thisMessage = cursor.getString(cursor.getColumnIndex("_id"));
            if (type.equals("2")&&(!
(MainActivity.thisMessage.equals(MainActivity.lastMessage)))){
               if (MainActivity.sms>1) {
                  MainActivity.sms=MainActivity.sms-1;
                  if ((MainActivity.sms<=10)&&(MainActivity.sms>1)){
                     Toast.makeText(MainActivity.context, "Running Out Of free SMS",
Toast.LENGTH_LONG).show();
                  }
                  if (MainActivity.sms==0){
                     Toast.makeText(MainActivity.context, "You just run out of Free
SMS",Toast.LENGTH_LONG).show();
                  }
               }else if(MainActivity.sms==0){
                  Toast.makeText(MainActivity.context, "You have run out of Free
SMS",Toast.LENGTH_LONG).show();
               }
               MainActivity.sms=Math.max(MainActivity.sms,0);
               MainActivity.smsPercent=(100*MainActivity.sms)/MainActivity.smsMax;
               MainActivity.lastMessage = MainActivity.thisMessage;
            }
         }
         catch (CursorIndexOutOfBoundsException e){
         }
         finally{
            cursor.close();
         }

         if (outgoingSMS != null ){
          Log.d("OBSERVER", "SMSHelper: outgoingSMS: " + outgoingSMS);
         }
      }
```

```
    }
}
```

A.2.5. SetMax.java

```
package kozos.trindicator;

import android.app.Activity;
import android.app.DialogFragment;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.LayoutInflater;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class SetMax extends DialogFragment implements View.OnClickListener {
    Button SetButton,CancelButton;
    Communicator comm;
    public void onAttach(Activity a){
        super.onAttach(a);
        comm=(Communicator)a;
    }
    @Override
    public View onCreateView(LayoutInflater inflater,ViewGroup container,Bundle
savedInstanceState){
        View view;
        view=inflater.inflate(R.layout.set_max_dialog,null);
        SetButton=(Button) view.findViewById(R.id.setButton);
        CancelButton=(Button) view.findViewById(R.id.cancelButton);
        SetButton.setOnClickListener(this);
        CancelButton.setOnClickListener(this);
        setCancelable(false);
        return view;
```

```
    }
    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        TextView tv;
        switch (MainActivity.maxSwitch){
            case 1:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
                tv.setText("SET MINUTES MAXIMUM VALUE");
                tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
                tv.setText(Integer.toString(MainActivity.minutesMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
                tv.setText("Current minutes maximum:
"+Integer.toString(MainActivity.minutesMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentValue);
                tv.setText("Current minutes value: "+Integer.toString(MainActivity.minutes));
                break;

            case 2:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
                tv.setText("SET SMS MAXIMUM VALUE");
                tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
                tv.setText(Integer.toString(MainActivity.smsMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
                tv.setText("Current SMS maximum: "+Integer.toString(MainActivity.smsMax));
                tv = (TextView)getView().findViewById(R.id.textCurrentValue);
                tv.setText("Current SMS value: "+Integer.toString(MainActivity.sms));
                break;

            case 3:
                tv = (TextView)getView().findViewById(R.id.textMaximumHeader);
                tv.setText("SET DATA MAXIMUM VALUE");
                tv = (TextView)getView().findViewById(R.id.textMaximumSetter);
```

```
            tv.setText(Integer.toString(MainActivity.dataMax));
            tv = (TextView)getView().findViewById(R.id.textCurrentMaximum);
            tv.setText("Current data maximum: "+Integer.toString(MainActivity.dataMax));
            tv = (TextView)getView().findViewById(R.id.textCurrentValue);
            tv.setText("Current data value: "+Integer.toString(MainActivity.data));
            break;
        }
    }

    @Override
    public void onClick(View v) {
        int setvalue=0;

        TextView tv;
        if(v.getId()==R.id.setButton){
            tv =(TextView) getView().findViewById(R.id.textMaximumSetter);
            if (!(tv.getText().length()==0)) {

                setvalue = Integer.parseInt(tv.getText().toString());
                if (MainActivity.maxSwitch == 1) {
                    if ((setvalue<1)||(setvalue<MainActivity.minutes)){
                        setvalue=100;
                        Toast.makeText(MainActivity.context, "Maximum must be positive and at
least equal to current value",Toast.LENGTH_LONG).show();}
                    MainActivity.minutesMax = Math.max(setvalue,MainActivity.minutes);

                } else if (MainActivity.maxSwitch == 2) {
                    if ((setvalue<1)||(setvalue<MainActivity.sms)){
                        setvalue=100;
                        Toast.makeText(MainActivity.context, "Maximum must be positive and at
least equal to current value",Toast.LENGTH_LONG).show();}
                    MainActivity.smsMax = Math.max(setvalue,MainActivity.sms);
```

```
            } else {
                if ((setvalue<1)||(setvalue<MainActivity.data)){
                    setvalue=100;
                    Toast.makeText(MainActivity.context, "Maximum must be positive and at
least equal to current value",Toast.LENGTH_LONG).show();}
                MainActivity.dataMax =Math.max (setvalue,MainActivity.data);
            }
            MainActivity.maxSwitch = 0;
            dismiss();
          }else{
            dismiss();
            Toast.makeText(MainActivity.context, "wrong
form",Toast.LENGTH_LONG).show();
          }
          comm.Respond("Update");
       }
       else {
          dismiss();
       }
    }
}
```

A.2.6. Communicator.java (Interface)

```
 interface Communicator{
    public void Respond(String data);
}
```

A.2.7. SMSOutgoing.java

```
package kozos.trindicator;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
```

```
import android.widget.Toast;

public class SMSOutgoing extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent) {
      Log.d("Log", "BR Sent SMS 1");
   if (MainActivity.sms>1) {
   MainActivity.sms=MainActivity.sms-1;
      Log.d("Log", "BR REDUCE SMS");
   if ((MainActivity.sms<=10)&&(MainActivity.sms>1)){
      Toast.makeText(MainActivity.context, "Running Out Of free SMS",
Toast.LENGTH_LONG).show();
   }
   if (MainActivity.sms==0){
      Toast.makeText(MainActivity.context, "You just run out of Free
SMS",Toast.LENGTH_LONG).show();
   }
   }else if(MainActivity.sms==0){
      Toast.makeText(MainActivity.context, "You have run out of Free
SMS",Toast.LENGTH_LONG).show();
   }
   MainActivity.sms=Math.max(MainActivity.sms,0);
   MainActivity.smsPercent=(100*MainActivity.sms)/MainActivity.smsMax;
   }
}
```

A.2.8. SMSReceiver.java

```
package kozos.trindicator;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.telephony.SmsMessage;
import android.widget.Toast;
import java.util.Date;

public class SMSReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent)
   {
      String[] strArray;
      Bundle bundle = intent.getExtras();
      SmsMessage[] msgs = null;
      String str = "";
      if (bundle != null) {
         Object[] pdus = (Object[]) bundle.get("pdus");
         msgs = new SmsMessage[pdus.length];
         for (int i = 0; i < msgs.length; i++) {
            msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
            str += msgs[i].getMessageBody().toString();
            Toast.makeText(context, msgs[0].getOriginatingAddress().toString() + "  " + str,
Toast.LENGTH_SHORT).show();
            strArray = str.split(" ");
            if (((msgs[0].getOriginatingAddress().equals("XXXXXXXXXX") ||
msgs[0].getOriginatingAddress().equals("+30XXXXXXXXXX"))) &&
strArray[0].equals("0")) {
               MainActivity.minutes = Integer.parseInt(strArray[1]);

MainActivity.minutesMax=Math.max(MainActivity.minutes,MainActivity.minutesMax);
               MainActivity.sms = Integer.parseInt(strArray[2]);
               MainActivity.smsMax=Math.max(MainActivity.sms,MainActivity.smsMax);
               MainActivity.data = Integer.parseInt(strArray[3]);
               MainActivity.dataMax=Math.max(MainActivity.data,MainActivity.dataMax);
               MainActivity.lastUpdate=new Date();
```

```
MainActivity.UpdateDate=MainActivity.formatter.format(MainActivity.lastUpdate);
                MainActivity.hasUpdated=true;
                MainActivity.previousTotalBytes=MainActivity.res;
                MainActivity.lastUpdateMS=System.currentTimeMillis();
                }
            }
        }
    }
  }
```

A.3. To manifest (Στο directory: App/manifests/)

A.3.1. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="kozos.trindicator" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/trindicatoricon"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".MainActivity"
         android:label="@string/app_name"
         android:launchMode="singleInstance" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
```

```
      <receiver android:name="kozos.trindicator.PhoneStateBroadcastReceiver">
         <intent-filter>
            <action android:name="android.intent.action.PHONE_STATE">
            </action></intent-filter>

      </receiver>
      <receiver android:name="kozos.trindicator.SMSReceiver">
         <intent-filter>
            <action android:name=
               "android.provider.Telephony.SMS_RECEIVED" />
         </intent-filter>
      </receiver>
      <receiver android:name="kozos.trindicator.SMSOutgoing">
         <intent-filter>
            <action android:name=
               "android.provider.Telephony.SMS_SENT" />
         </intent-filter>
      </receiver>
   </application>
   <uses-permission android:name="android.permission.READ_PHONE_STATE">
   </uses-permission>
   <uses-permission android:name="android.permission.READ_SMS"></uses-permission>
   <uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
   <uses-permission android:name="android.permission.RECEIVE_SMS">
   </uses-permission>
   <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

Α.4. Τα layout των όψεων της εφαρμογής (Στο directory: /res/layout/)

Α.4.1. activity_main.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    android:fillViewport="true"
    >
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:background="#214193"
    tools:context=".MainActivity">

    <com.github.lzyzsd.circleprogress.ArcProgress
        android:id="@+id/arcMinutes"
        android:background="#214193"
        android:layout_width="150dp"
        android:layout_height="150dp"
        custom:arc_progress="55"
        custom:arc_bottom_text="Minutes"
        custom:arc_bottom_text_size="17dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <com.github.lzyzsd.circleprogress.ArcProgress
        android:id="@+id/arcSMS"
        android:background="#214193"
        android:layout_width="150dp"
        android:layout_height="150dp"
        custom:arc_progress="25"
```

```
    custom:arc_bottom_text="SMS"
    custom:arc_bottom_text_size="17dp"
    android:layout_below="@+id/arcMinutes"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="45dp" />

<com.github.lzyzsd.circleprogress.ArcProgress
    android:id="@+id/arcData"
    android:background="#214193"
    android:layout_width="150dp"
    android:layout_height="150dp"
    custom:arc_progress="85"
    custom:arc_bottom_text="Data"
    custom:arc_bottom_text_size="17dp"
    android:layout_below="@+id/arcSMS"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="45dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="    55 / 100   "
    android:id="@+id/textMinutesDisplay"
    android:layout_marginTop="7dp"
    android:gravity="center"
    android:layout_alignParentTop="true"
    android:textColor="#3D8FF4"
    android:layout_toRightOf="@+id/arcMinutes"
    android:layout_toEndOf="@+id/arcMinutes"
    android:layout_marginLeft="40dp"
```

```
    android:layout_marginStart="7dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="  Toggle Notification  "
    android:id="@+id/toggleMinutesButton"
    android:onClick="ToggleMinutesNotification"
    style="?android:attr/borderlessButtonStyle"
    android:textColor="#3D8FF4"
    android:background="#4668B2"
    android:layout_below="@+id/textMinutesDisplay"
    android:layout_toRightOf="@+id/arcMinutes"
    android:layout_toEndOf="@+id/arcMinutes"
    android:layout_marginTop="15dp"
    android:layout_marginLeft="15dp"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="    25 / 100   "
    android:gravity="center"
    android:id="@+id/textSMSDisplay"
    android:textColor="#3D8FF4"
    android:layout_alignTop="@+id/arcSMS"
    android:layout_alignLeft="@+id/textMinutesDisplay"
    android:layout_alignStart="@+id/textMinutesDisplay"
    android:layout_marginTop="7dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="  Toggle Notification  "
    android:id="@+id/toggleSMSButton"
    style="?android:attr/borderlessButtonStyle"
    android:onClick="ToggleSMSNotification"
    android:textColor="#3D8FF4"
    android:background="#4668B2"
    android:layout_centerVertical="true"
    android:layout_below="@+id/textSMSDisplay"
    android:layout_alignLeft="@+id/toggleMinutesButton"
    android:layout_alignStart="@+id/toggleMinutesButton"
    android:layout_marginTop="15dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="    85 / 100   "
    android:id="@+id/textDataDisplay"
    android:gravity="center"
    android:textColor="#3D8FF4"
    android:layout_marginTop="7dp"
    android:layout_alignTop="@+id/arcData"
    android:layout_alignLeft="@+id/textSMSDisplay"
    android:layout_alignStart="@+id/textSMSDisplay" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="  Toggle Notification  "
    android:id="@+id/toggleDataButton"
    android:onClick="ToggleDataNotification"
    android:textColor="#3D8FF4"
```

```
    android:background="#4668B2"
    style="?android:attr/borderlessButtonStyle"
    android:layout_below="@+id/textDataDisplay"
    android:layout_alignLeft="@+id/toggleSMSButton"
    android:layout_alignStart="@+id/toggleSMSButton"
    android:layout_marginTop="15dp"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text=" Last Update: 18:44 17/12/2015 "
    android:id="@+id/textLastUpdate"
    android:layout_below="@+id/arcData"
    android:layout_alignParentLeft="true"
    android:gravity="center"
    android:layout_alignParentStart="true"
    android:textColor="#3D8FF4"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Update"
    android:id="@+id/updateButton"
    android:onClick="SMSClick"
    android:textColor="#3D8FF4"
    android:background="#4668B2"
    android:layout_below="@+id/textLastUpdate"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="15dp"
    style="?android:attr/borderlessButtonStyle"/>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Help"
    android:id="@+id/helpButton"
    android:onClick="HelpDialog"
    android:textColor="#3D8FF4"
    android:background="#4668B2"
    android:layout_below="@+id/updateButton"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="15dp"
    style="?android:attr/borderlessButtonStyle"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Set Max"
    android:id="@+id/setMinutesMaxButton"
    android:onClick="SetMaxDialog"
    android:textColor="#3D8FF4"
    android:background="#4668B2"
    style="?android:attr/borderlessButtonStyle"
    android:layout_below="@+id/toggleMinutesButton"
    android:layout_toRightOf="@+id/updateButton"
    android:layout_toEndOf="@+id/updateButton"
    android:layout_marginTop="15dp"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Set Max"
    android:id="@+id/setSMSMaxButton"
```

```
        android:onClick="SetMaxDialog"
        android:layout_below="@+id/toggleSMSButton"
        android:layout_alignLeft="@+id/setMinutesMaxButton"
        android:layout_alignStart="@+id/setMinutesMaxButton"
        android:textColor="#3D8FF4"
        android:background="#4668B2"
        style="?android:attr/borderlessButtonStyle"
        android:layout_marginTop="15dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Set Max"
        android:id="@+id/setDataMaxButton"
        android:onClick="SetMaxDialog"
        android:layout_below="@+id/toggleDataButton"
        android:layout_alignLeft="@+id/setMinutesMaxButton"
        android:layout_alignStart="@+id/setMinutesMaxButton"
        android:textColor="#3D8FF4"
        android:background="#4668B2"
        style="?android:attr/borderlessButtonStyle"
        android:layout_marginTop="15dp" />



</RelativeLayout>
 </ScrollView>
```

A.4.2. help_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="none"
        android:focusable="false"
        android:ems="10"
        android:maxLines="12"
        android:scrollbars="vertical"
        android:id="@+id/helpText"
        android:text="Main Screen\n\nThe main screen consists of three groups of displays and buttons,one for each monitored value.Additional information and functionalities is provided below them.\nEach one of the three groups consists of an Arc-gauge a text field and two buttons. The arc-gauge provides an easy to reference expression of the remaining value,also expressed as a percentage in the center of it. Under the arc the name of each measured value is provided. The information is also displayed as a fraction of the current value to the maximum/reference value. To change the maximum/reference value tap the SET MAX button,for further instructions please look under Set Max Screen. The TOGGLE NOTIFICATION button creates or removes a notification regarding the value.\nBeneath the three grouped displays infromation regarding the last time the values were synchronized with the provider is also displayed. Tap on the UPDATE button to initiate synchronization manually.\nFinally,a HELP button is provided that brings forth this screen.\n\nSet Max Screen\n\nThrough this screen you can change the max/reference value of a measured value. The current value and the current maximum value are also provided. Tap on the CANCEL button to abort the screen without changes and return to the Main Screen or tap on the SET MAX button to set the number in the top text field as maximum/reference value." />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK"
        android:id="@+id/OkHelpButton"
        android:layout_below="@+id/helpText"
        android:layout_marginTop="15dp"
```

```
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

A.4.3 set_max_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textMultiLine"
    android:ems="10"
    android:editable="false"
    android:id="@+id/textMaximumHeader"
    android:layout_alignParentTop="true"
    android:text="SET XXXXXXX MAXIMUM VALUE"
    android:gravity="center"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Current XXXXXX Maximum: 6787867"
    android:id="@+id/textCurrentMaximum"
    android:layout_marginTop="29dp"
    android:gravity="center"
    android:layout_below="@+id/textMaximumSetter"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
```

```
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Current XXXXXX Value: 78787"
    android:id="@+id/textCurrentValue"
    android:layout_marginTop="29dp"
    android:layout_below="@+id/textCurrentMaximum"
    android:gravity="center"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/textMaximumSetter"
    android:layout_below="@+id/textMaximumHeader"
    android:text="789829"
    android:layout_alignParentLeft="true"
    android:gravity="center"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<Button
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Set Max"
        android:id="@+id/setButton"
        android:layout_marginTop="27dp"
        android:layout_below="@+id/textCurrentValue"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="53dp"
        android:layout_marginStart="53dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:id="@+id/cancelButton"
        android:layout_alignTop="@+id/setButton"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginRight="53dp"
        android:layout_marginEnd="53dp" />
</RelativeLayout>
```

Βιβλιογραφία

1. Eckel Bruce,"Thinking in Java 4th Edition",Prentice Hall PTR, 2006.

2. Smyth Neil, "Android Studio Development Essentials", eBookFrenzy, 2015.

3. Griffiths David, Griffiths Dawn, "Head First Android Development", O'Reilly, 2015.

4. Alan Dix Janet Finlay, Gregory D. Abowd, Beale Russell ,
   Επικοινωνία Ανθρώπου-Υπολογιστή, Μ. Γκιούρδας 2012.

5. Christopher D. Wickens, John D. Lee, Yili Liu, Sallie Gordon-Becker.
   Introduction to Human Factors Engineering, Pearson, 2003 (2nd Edition)

6. "Java 7 Documentation",
http://www.oracle.com/technetwork/java/javase/overview/index.html

7. lzyzsd, "CircleProgress:CircleProgress, DonutProgress, ArcProgress",
https://github.com/lzyzsd/CircleProgress

8. Χαρακτηριστικά και Αρχιτεκτονική του Android
http://www.tutorialspoint.com/android/android_overview.htm
http://www.tutorialspoint.com/android/android_architecture.htm
http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

9. Αλληλεπίδραση Ανθρώπου-Υπολογιστή
https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction

10. Τα εικονίδια της εφαρμογής ελήφθησαν από το www.flaticon.com και σχεδιάστηκαν από τον Freepik.